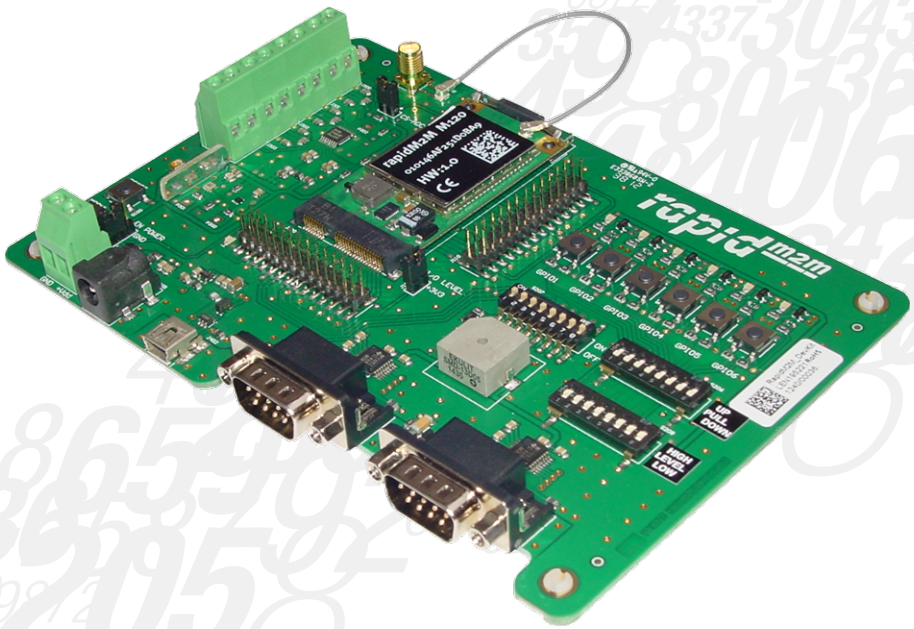
User manual for the rapidM2M developer package M120



Rev. 03 valid from:

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

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

6872
6872
6872

Chapter 1 Table of contents

User manual for the rapidM2M developer package M120 ... 1

Chapter 1 Table of contents ... 3

Chapter 2 Technical data ... 15

Chapter 3 General specifications ... 19

3.1 General product information ... 19

3.2 Scope of supply ... 19

3.3 Dimensions ... 20

3.4 Device labelling ... 20

3.5 Storage of the product ... 21

Chapter 4 Tutorial ... 23

4.1 Description of the "app.p" script function ... 23

4.2 Prerequisites ... 24

4.3 Step-by-step instructions ... 24

4.4 Explanation of the rapidM2M specific code blocks of the "app.p" script ... 45

4.4.1 Initialising the sensors ... 45

4.4.2 Initialising the timers ... 46

4.4.3 Determining the callback for handling the configuration ... 46

4.4.4 Setting the connection type ... 46

4.4.5 Saving measurement data ... 47

4.4.6 Monitoring button S201 to initiate an immediate recording ... 47

4.4.7 Reading the sensor values ... 48

4.4.8 Reading the configuration ... 49

4.5 Explanation of the connector definition included in the "app_connector.txt" file ... 50

4.5.1 Split tag ... 50

4.5.2 Measurement data channel o ... 51

4.5.3 Configuration memory block o ... 52

Chapter 5 Functional principle ... 55

5.1 Recommended procedure ... 58

5.1.1 Development of M2M/IoT application ... 58
5.2 Automatic selection of the GSM network ... 59
5.2.1 Ban list ... 59
5.3 Procedure in case of connection aborts ... 60
5.3.1 Connection abort in "online" mode ... 61
5.3.2 Connection abort during a script download ... 61
5.4 Error handling ... 62
5.5 Registration memory blocks ... 62
5.5.1 REG_SYS_FLASH ... 63
5.5.2 REG_APP_OTP ... 63
5.6 File transfer ... 64
5.7 Meaning of the SIM state ... 64
Chapter 6 rapidM2M board for developers ... 67
6.1 Overview ... 67
6.1.1 Block diagram ... 70
6.1.2 RS232 interfaces ... 71
6.1.3 Simulation circuits ... 72
6.1.3.1 Output of digital signals ... 72
6.1.3.2 Input of digital signals ... 73
6.1.4 Jumper for supplying the level shifters (J111) ... 75
6.1.5 Jumper for selecting the USB supply for the module (J113) ... 75
6.1.6 Plug connector to which the odd-numbered pins of the M120 module are connected (J106 and J109) ... 75
6.1.7 Plug connector to which the even-numbered pins of the M120 module are connected (J107 and J110) ... 77
6.1.8 USB interface ... 79
6.1.9 Status LED (LED100) ... 79
6.1.10 Supply (J105 or J108) ... 79
6.1.10.1 Terminal strip (J108) ... 79
6.1.10.2 DC coaxial connector (J105) ... 80

6.1.11 Jumper to activate the energy supply of the M120 module (J104) 80
6.1.12 Button to activate the wireless GSM controller on the rapidM2M module M120 (S101) 80
6.1.13 Jumper to switch on the wireless GSM controller on the rapidM2M module M120 (J37) 80
6.1.14 Button to reset the wireless GSM controller on the M120 module (S100) 81
6.1.15 Sensors 81
6.1.15.1 ADC AD7792 (U302) 82
6.1.15.2 Temperature sensor TMP112 (U300) 83
6.1.15.3 Ambient light and IR proximity sensor MAX44000 (U301) 84
6.1.16 Holding fixture for the M120 module 84
6.2 Circuit diagram 85
6.2.1 Overview 85
6.2.2 Connector 86
6.2.3 USB RS232 87
6.2.4 Sensors 88
6.2.5 GPIO 89
6.3 Layout 90
6.4 Electrical installation 91
6.4.1 Information on preventing electrostatic discharges (ESD) 91
6.4.2 Inserting the M120 module 92
6.4.3 Connecting the sensors, actuators and power supply 93
6.4.4 Connecting the GSM antenna 95
Chapter 7 rapidM2M module M120 97
7.1 Technical data 97
7.2 Overview 99
7.2.1 Block diagram 100
7.2.2 52pin connector strip (X2) 100
7.2.3 Memory organisation 103
7.2.4 Device labelling 104

7.3 Dimensions ... 104
7.3.1 Dimensions of the 52pin connector strip (X2) ... 105
7.4 Technical details ... 105
7.4.1 Technical details about the energy supply ... 105
7.4.1.1 Supply via VIN ... 106
7.4.1.2 Supply via VAUX ... 106
7.4.2 ENPower signal ... 106
7.4.3 ENCPU signal ... 106
7.4.4 Reset signal ... 107
7.4.5 Status LED ... 107
7.4.6 Technical details about the level shifters ... 107
7.4.7 Serial interfaces ... 108
7.4.7.1 UART ... 108
7.4.7.1.1 Pin description ... 108
7.4.7.2 SPI ... 109
7.4.7.2.1 Pin description ... 109
7.4.7.2.2 SPI waveforms ... 110
7.4.7.2.3 Application ... 110
7.4.7.3 I2C ... 110
7.4.7.3.1 Pin description ... 110
7.4.7.3.2 I2C waveforms ... 111
7.4.7.3.3 Application ... 111
7.4.8 GPIO ... 112
7.4.9 Interrupt inputs ... 112
7.5 Typical application ... 112
7.5.1 Supply via VAUX ... 112
7.5.2 Supply via VIN ... 113
7.5.3 Valid for both applications ... 114
Chapter 8 Cloud server ... 115
8.1 New registration and logging in ... 115

8.1.1 Creating a new account ... 116
8.1.2 Login ... 116
8.2 Overview ... 117
8.2.1 Explanation of the symbols ... 117
8.3 "Sites" area ... 118
8.3.1 Reports ... 119
8.3.1.1 Basic configuration ... 119
8.3.2 Accounts ... 120
8.3.3 Sites ... 121
8.3.3.1 Creating a new site ... 123
8.3.3.2 Site position ... 124
8.3.3.3 List of files ... 125
8.3.3.3.1 Editing the file ... 126
8.3.3.4 Positioning the site on the map ... 127
8.4 "User" area ... 128
8.4.1 Creating users ... 128
8.5 "Alarms" area ... 129
8.5.1 "Alarms" sub-section ... 130
8.5.2 "Acknowledged alarms" sub-section ... 131
8.5.3 "Alarm archive" sub-section ... 132
8.5.4 "Alarm schedules" sub-section ... 133
8.5.4.1 Alarm schedule and comment ... 133
8.5.4.2 Setup for the alarm by telephone ... 133
8.5.5 Notifications ... 133
8.6 "Statistics" area ... 134
8.6.1 "SMS/voice calls" sub-section ... 134
8.6.2 "SIM" sub-section ... 135
8.7 "My data" area ... 136
8.8 Application templates of the Cloud server ... 136
Chapter 9 rapidM2M Toolset ... 137

9.1 General ... 137
9.2 Prerequisites ... 137
9.3 Functioning principle ... 138
9.4 Installation ... 139
9.5 Overview ... 142
9.6 "Script" tab ... 144
9.7 "Config" tab ... 146
9.8 "Console" tab ... 147
9.9 "COM Data" tab ... 147
9.10 "Server" tab ... 149
9.11 "Log" tab ... 149
9.12 "FW" tab ... 151
9.13 Hot keys ... 152
Chapter 10 XML interface ... 153
10.1 General ... 153
10.2 Structure of the database ... 153
10.2.1 Generally applicable table identifiers ... 156
10.2.1.1 rm2mid ... 156
10.2.1.2 geraetid ... 159
10.2.1.3 bookingline (read only) ... 161
10.2.2 Device-specific table identifier ... 162
10.2.2.1 histdata0 - histdata9 ... 162
10.2.2.2 rm2mlog ... 162
10.2.2.3 rm2mstate ... 162
10.2.2.4 rm2mlocals ... 163
10.2.2.5 rm2mpos ... 163
10.2.2.6 rm2m_ctrlscript ... 164
10.2.2.7 reg_sys_otp ... 165
10.2.2.8 reg_sys_flash ... 165
10.2.2.9 reg_app_otp ... 165

10.2.2.10 reg_app_flash ... 166
10.2.2.11 configo - config9 ... 166
10.2.2.12 rm2mraw ... 166
10.3 XML export ... 166
10.3.1 Configuration of a HTTP-GET command ... 166
10.3.1.1 Syntax of the xPath query ... 167
10.3.1.1.1 List of the fields in a table ... 167
10.3.1.1.2 List of site types ... 168
10.3.1.1.3 Filter conditions ... 168
10.3.1.1.3.1 Example: ... 169
10.3.1.1.4 Fields ... 170
10.3.1.1.4.1 Example ... 170
10.3.1.1.5 Data retrieval including index specifications ... 170
10.3.1.1.5.1 Example ... 170
10.3.1.2 Data record compression ... 172
10.3.1.3 Updating data fields ... 174
10.3.1.3.1 Example ... 174
10.3.1.4 Data retrieval via streaming technology ... 175
10.3.1.4.1 Events ... 176
10.3.1.5 Output formats of the HTTP-GET command ... 177
10.3.1.5.1 Format of the XML result ... 177
10.3.1.5.1.1 Example: ... 178
10.3.1.5.2 Format of the JSON result ... 178
10.3.1.5.2.1 Example ... 179
10.4 XML import ... 179
10.4.1 Transferring data to the Cloud server ... 179
10.4.1.1 Configuration of a HTTP-POST command ... 180
10.4.2 Structure of the XML file ... 181
10.4.2.1 Changing existing data ... 181
10.4.2.2 Importing data ... 182

Chapter 11 Placing into operation ... 185
11.1 Placing the system into operation ... 185
11.2 Testing communication with the device ... 185
Chapter 12 User interface on the Cloud-Server ... 189
12.1 Site configuration ... 189
12.1.1 Site ... 189
12.1.2 Comments ... 189
12.1.3 Control ... 189
12.1.4 Configuration 0 - configuration 1 ... 190
12.1.5 Alarm settings ... 190
12.1.6 Basic setting ... 190
12.1.7 FTP export settings ... 190
12.2 Device configuration ... 191
12.2.1 Comments ... 191
12.2.2 Measurement instrument ... 191
Chapter 13 Pawn script ... 193
13.1 General ... 193
13.1.1 Direct input of a pawn script ... 193
13.1.2 Uploading a binary file ... 193
13.1.3 Script editor of the rapidM2M Toolset ... 193
13.2 rapidM2M API ... 194
13.2.1 Core functions ... 194
13.2.2 Console functions ... 197
13.2.3 rapidM2M functions ... 198
13.2.3.1 Arrays with symbolic indices ... 198
13.2.3.2 Constants ... 200
13.2.3.3 Callback functions ... 203
13.2.3.4 Functions ... 204
13.2.4 File transfer functions ... 236
13.2.4.1 Arrays with symbolic indices ... 236

13.2.4.2 Constants ... 236
13.2.4.3 Callback functions ... 237
13.2.4.4 Functions ... 238
13.2.5 Mathematical functions ... 241
13.2.6 String functions ... 243
13.2.7 Help functions ... 248
13.2.7.1 Arrays with symbolic indices ... 248
13.2.7.2 Constants ... 248
13.2.7.3 Functions ... 248
13.3 Pawn script error codes ... 251
13.4 Syntax ... 253
13.4.1 General syntax ... 253
13.4.1.1 Format ... 253
13.4.1.2 Optional semicolons ... 253
13.4.1.3 Comments ... 253
13.4.1.4 Identifier ... 253
13.4.1.5 Reserved keywords ... 254
13.4.1.6 Numerical constants ... 254
13.4.1.6.1 Numerical integer constants ... 254
13.4.1.6.2 Numerical floating-point constants ... 254
13.4.2 Variables ... 255
13.4.2.1 Declaration ... 255
13.4.2.2 Local declaration ... 255
13.4.2.3 Global declaration ... 255
13.4.2.4 Static local declaration ... 255
13.4.2.5 Static global declaration ... 255
13.4.2.6 Floating point values ... 256
13.4.3 Constant variables ... 256
13.4.4 Array variables ... 256
13.4.4.1 One-dimensional arrays ... 256

13.4.4.2 Initialisation 256
13.4.4.3 Progressive initialisation for arrays 257
13.4.4.4 Multi-dimensional arrays 257
13.4.4.5 Arrays and the "sizeof" operator 258
13.4.5 Operators and expressions 258
13.4.5.1 Notational conventions 258
13.4.5.2 Expressions 259
13.4.5.3 Arithmetic 259
13.4.5.4 Bit manipulation 260
13.4.5.5 Assignment 260
13.4.5.6 Comparative operators 261
13.4.5.7 Boolean 261
13.4.5.8 Other 262
13.4.5.9 Priority of the operators 262
13.4.6 Statements 263
13.4.6.1 Statement label 264
13.4.6.2 Composite statements 264
13.4.6.3 Expression statement 264
13.4.6.4 Empty statement 264
13.4.6.5 Assert expression 264
13.4.6.6 Break 265
13.4.6.7 Continue 265
13.4.6.8 Do statement while (expression) 266
13.4.6.9 Exit expression 266
13.4.6.10 For (expression 1; expression 2; expression 3) statement 266
13.4.6.11 Goto label 267
13.4.6.12 If (expression) statement 1 else statement 2 267
13.4.6.13 Return expression 268
13.4.6.14 switch (expression) {case list} 268
13.4.6.15 While (expression) statement 269

13.4.7 Functions ... 269
13.4.7.1 Function arguments ("call-by-value" versus "call-by-reference") ... 270
13.4.7.2 Named parameters versus fixed parameters ... 272
13.4.7.3 Standard values of function arguments ... 273
13.5 Example ... 274
13.6 Differences to C ... 274
Chapter 14 myDatanet Connector ... 277
14.1 Prerequisites ... 277
14.2 Split tag ... 278
14.3 Data structure ... 280
14.3.1 Attributes of the field definition ... 281
14.4 Example ... 284
14.5 Special values of the data types ... 285
Chapter 15 Troubleshooting and repair ... 287
15.1 General problems ... 287
15.2 Log entries and error codes ... 289
15.2.1 Modem error ... 292
Chapter 16 Spare parts and accessories ... 297
16.1 rapid M2M module ... 297
16.2 Antennas ... 297
16.3 Power supply ... 297
16.4 Adapter ... 297
16.5 Other accessories ... 297
Chapter 17 Document history ... 299
Chapter 18 Glossary ... 305
Chapter 19 Contact information ... 307

6872
6872
6872

Chapter 2 Technical data

Note: *The following technical specifications apply to the rapidM2M board for developers with the rapidM2M module M120 installed. A detailed technical description of the M120 module is provided in chapter "Technical data" on page 97.*

Voltage supply	5...32VDC , max. t.b.d. mA@12V
Housing	Weight: 134g (incl. rapidM2M module M120) Dimensions (WHD): 160 x 25 x 110mm (without antenna and connection cable)
Operating temperature	-5...+40°C, 15...90%rH non-condensing
Storage temperature	-20...+70°C
Antenna connector	SMA
Serial interfaces	2 x RS232[1)] for the data communication • Baud rate: 300-115200 • Stop bits: 1, 2 • Parity: N, E, O • Data bits: 7, 8 1 x SPI • Mode: SPI master • Adjustable clock polarity/phase • Clock frequency: 102kHz...13MHz • Interface: 4-wire 1 x I²C • Mode: Master • Clock frequency: 100kHz...400kHz Additional information is provided in: • "RS232 interfaces" on page 71 • "SPI" on page 109 • "I2C" on page 110
USB interface	1 x mini-B USB 2.0 slave for establishing a connection to a PC. The rapidM2M Toolset must be installed on the PC to enable communication with the rapidM2M developer package M120 . Additional information is provided in "USB interface" on page 79.

Temperature sensor[2)]	Measurement range: -40...+125°C Resolution: 0,0625°C Additional information is provided in "Temperature sensor TMP112 (U300)" on page 83.
Ambient light and IR proximity sensor[2)]	Measurement range: 0,03...65535lux Additional information is provided in "Ambient light and IR proximity sensor MAX44000 (U301)" on page 84.
Analogue inputs[3)]	3 x differential inputs • Resolution: 16 Bit • Input voltage range: V_{ref} / gain • Internal reference voltage: 1,17V • External reference voltage: nominal 2,5V, max 3,3V • Gain: 1-128 Additional information is provided in "ADC AD7792 (U302)" on page 82.
Outputs[3)]	2 x constant current output • Output current: 10/210/1000µA Additional information is provided in "ADC AD7792 (U302)" on page 82.
I/O lines	6 x GPIO[4)] 2 x interrupt inputs
Simulation circuits	6 x circuits to input digital signals (button) or display digital signals (LED) 1 x buzzer to issue a digital signal
Data memory	3 x 349.525 Byte internal flash memory. The size of the data records is variable (max. 1024 Byte) and is determined by the script created by the user. The system-related overhead is 10 Byte per data record.
Configuration memory	10 independent blocks each with 10kB
Registration memory	4 blocks each with 1kB and pre-defined purposes for storing device-specific data Additional information is provided in "Registration memory blocks" on page 62.
Memory for the PAWN binary	256kB (uncompressed size) Additional information is provided in "Memory organisation" on page 103.
Data transmission	By means of GSM/GPRS quad-band modem to the respective Cloud server
SIM	The rapidM2M module M120 is equipped with an integrated SIM chip.

[1)]The RS232 level converters are located on the rapidM2M board for developers . During the subsequent integration of the M120 module in the customer-specific hardware, it must be taken into consideration that the output levels of the UARTs of the M120 module are dependent on the supply of the module's level shifters (see "Technical details about the level shifters" on page 107).

[2)]Realised via sensors (TMP112 and MAX44000), that are on the rapidM2M board for developers and are connected to the rapidM2M module M120 via I^2C bus.

[3)]Realised via an ADC (AD7792) that is on the rapidM2M board for developers and is connected to the rapidM2M module M120 via the SPI bus.

[4)]The 6 GPIOs can be connected to the simulation circuits via dip switch S206, S207 and S208.

6872
6872
6872

Chapter 3 General specifications

3.1 General product information

It is a developer package that is designed to help customers integrate the M120 module in their own devices. The rapidM2M board for developers is equipped with standardised connectors, plug connectors, terminal strips, buttons, LEDs (see "Overview" on page 67) as well as several sensors to ensure test circuits can be established easily and efficiently. The rapidM2M module M120 included in the scope of supply of the M120 developer package is a module for recording and transferring measurement data that is read-out via standardised microcontroller interfaces (2 x UART, 1 x SPI, 1 x I^2C). The customer can create his own application using a PAWN script (see "Pawn script" on page 193) and load it in rapidM2M module M120 via the USB interface or GPRS connection. This is either executed via rapidM2M Toolset (see "rapidM2M Toolset " on page 137) or via the XML interface of the server (see "XML interface" on page 153). The PAWN script enables access to the microcontroller interfaces thus providing the customer with the option to connect to almost all of the components that are compatible with these interfaces. The rapidM2M module M120 also comprises 6 GPIOs via which control signals can be exchanged between M120 and the surrounding hardware. 2 interrupt lines are provided for time-critical feedback from the surrounding hardware to the M120 .

The rapidM2M module M120 provides the customer with a memory area for his data (3 x 349.525 Byte) as well as 10 independent memory blocks each with 10kB for the configuration data. The rapidM2M module M120 also has 4 registration memory blocks each with 1kB and pre-defined purposes for storing device-specific data (see "Registration memory blocks" on page 62). The customer is responsible for initiating the transmission (see "rM2M_TxStart()"). The system automatically synchronises the configuration, registration and measurement data with the server. The M120 module is equipped with an integrated SIM chip for this purpose.

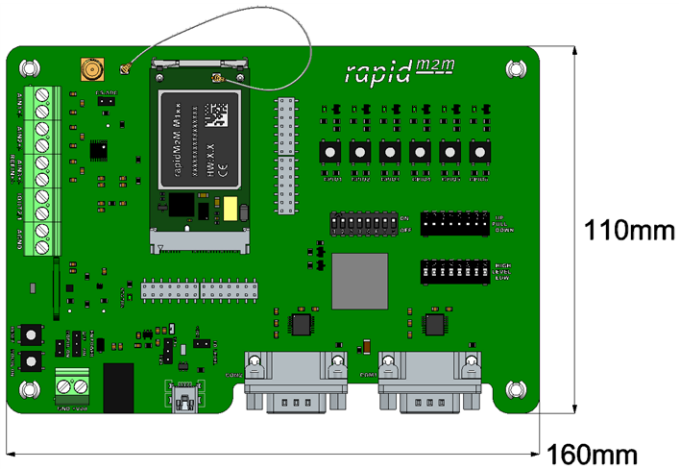
3.2 Scope of supply

The standard scope of supply of the developer package (300144) includes:

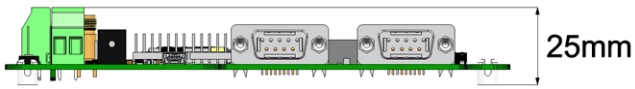
- rapidM2M board for developers
- rapidM2M module M120 (300087)
- Plug-in power supply 6W 12V
- Magnetic antenna SMA-M 2,5m (300085)
- 10 x jumpers
- U.FL antenna cable
- myDatanet Tool Pen (206.646)
- USB cable
- Null modem cable
- Quick Start (300414)
- Quick Guide (300384)

Additional accessories such as the M2M modules, antennas, power supplies, sensors, etc. are dependent on the order specifications. Please check these against the delivery slip.

3.3 Dimensions



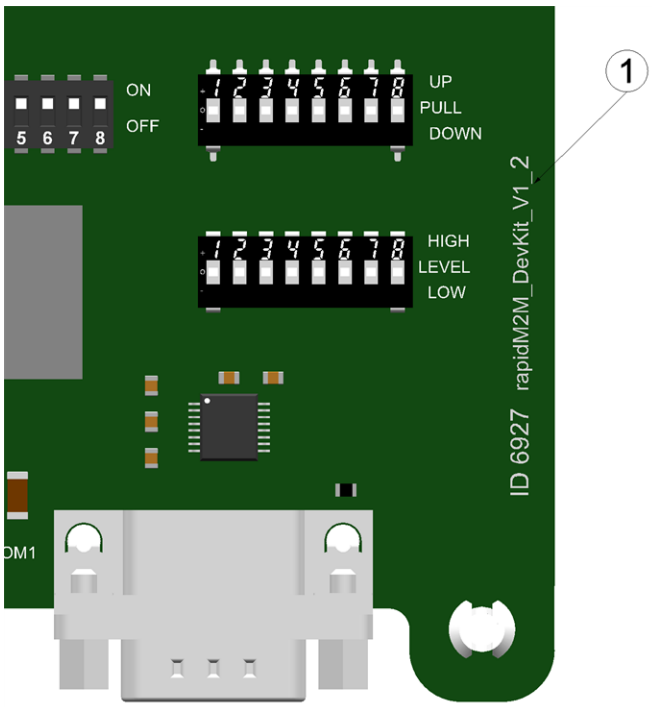
Dimensions: Width and depth



Dimensions: Height

3.4 Device labelling

The hardware version of the developer board is indicated on the right side of the circuit board near the S208 and S206 dip switches.



Hardware identification of the developer board

1 Hardware version	

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

3.5 Storage of the product

To safeguard the developer package, ensure that all of the relevant data was transferred to the Cloud server. If necessary, initiate a transmission directly on the device, if you have included this in your PAWN script, 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 Cloud server. This particularly applies to the "interval" connection type (see "rM2M_TxSetMode()"). If the "interval and wakeup" connection type has been selected, you can initiate the transmission via the Cloud 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 Cloud server. The data on the server is always up-to-date and the device can be switched off at any time. Then disconnect the rapidM2M board for developers from the supply voltage. If possible, switch off the power supply for all of the connected sensors before disconnecting the connection between the rapidM2M board for developers and the sensors. The antenna can then be removed. Store the rapidM2M developer package M120 including the accessories and connection cable in the original packaging.

The configuration and most recently determined data are retained. However, the system clock will not keep running without a power supply. This means that following recommissioning, the time must be synchronised to ensure that the measurement and log data timestamps are correct. This is completed automatically during the first connection to the Cloud server.

6872
6872
6872

Chapter 4 Tutorial

The following tutorial demonstrates how to create an M2M application using the "app.p" script provided with the rapidM2M Toolset , for example. You will also learn how to display data on the Cloud server using the reports and how you enable the input of configuration settings via the interface of the Cloud server.

4.1 Description of the "app.p" script function

The script records periodic measurement data and transfers it to the Cloud server. The interval for recording and transferring measurement data can be configured. The connection type can also be configured. If the "online" connection type is selected, the measurement data is transferred to the server as soon as the measurement data is created. The values from the temperature sensor TMP112 (U300) and the ambient light and IR proximity sensor MAX44000 (U301) are used in addition to some of the calculated measurement values. Both of the sensors are located on the rapidM2M board for developers and are connected to the rapidM2M module M120 via I^2C. An "alive" signal is issued via LED200 (GPIO1), which means that the LED flashes every second. The measurement values can be recorded immediately be pressing button S201 (GPIO2). LED202 - LED205 (GPIO2 - GPIO5) can be activated or deactivated via the configuration parameter using the interface of the Cloud server.

The script generates and transfers the following measurement values:

Measurement value	Unit	Explanation
Count	---	Infinite counter that contains the number of measurement data records generated since the last PowerOn
Time stamp	sec.	Seconds since 31.12.1999 (This is not the value that is used by the system to manage the measurement data.)
Temperature	°C	Measurement value read out via I^2C from temperature sensor TMP112 (U300) of the rapidM2M board for developers
Ambient light	lux	Measurement values read out via I^2C from ambient light and IR proximity sensor MAX44000 of the rapidM2M board for developers
Proximity	%	

The script provides the following configuration options:

Parameter	Unit	Explanation
Version	---	Display of the current version of the configuration (read only). Is specified by the script and should be changed if the structure of the configuration changes (e.g. addition of another parameter).
Record interval	sec.	Time between measurement data recordings
Transmission cycle	sec.	Time between transmissions
Mode	---	0: The device registers in the transmission cycle. 1: The device connects to the server in the transfer interval and can be placed into the ALOHA transmission mode via the server. 2: The device does not separate the connection and continuously transmits the measurement data.
GPIO ON/OFF	---	Enables the selective activation/deactivation of the LEDs. Bit0: LED202 (GPIO3); bit set -> LED ON Bit1: LED203 (GPIO4); bit set -> LED ON Bit2: LED204 (GPIO5); bit set -> LED ON Bit3: LED205 (GPIO6); bit set -> LED ON LED205 (GPIO6) lights up if this parameter is set to "8".

4.2 Prerequisites

You will require the following components for this tutorial:

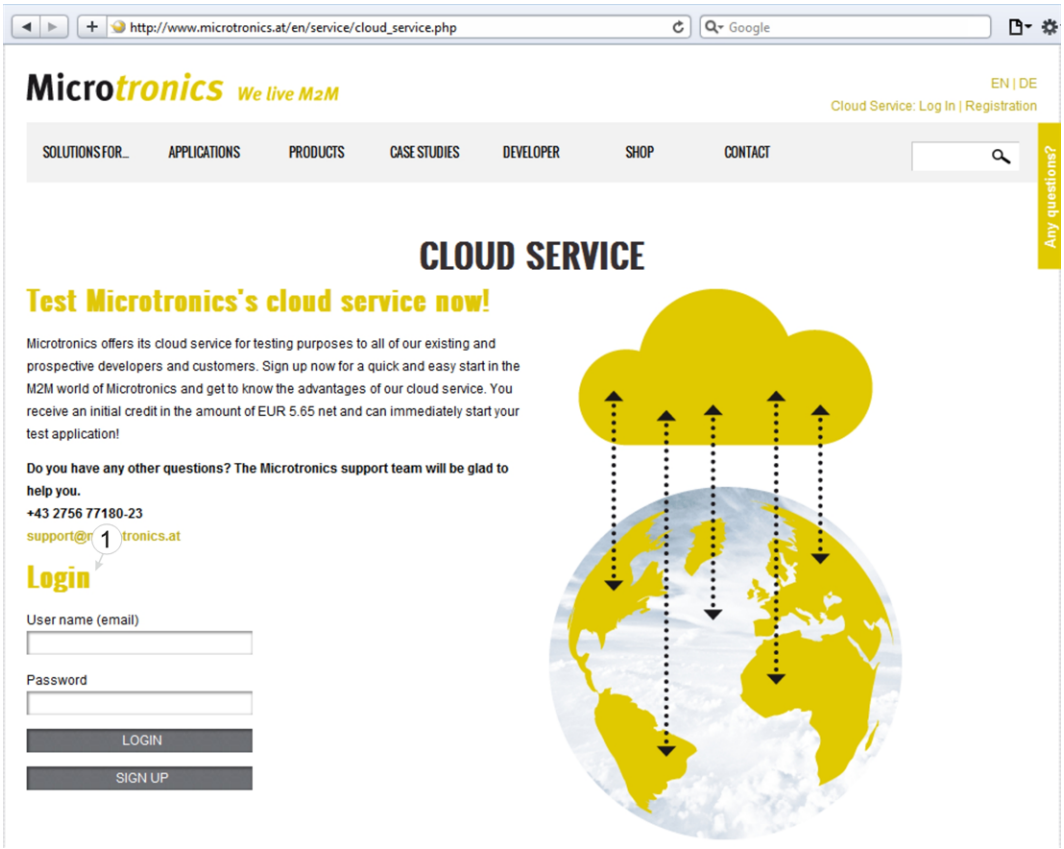
- PC with Microsoft operating system (Win XP, Windows Vista, Windows 7 or Windows 8)
- rapidM2M developer package M120 (300144)
- rapidM2M Toolset (minimum version 1.12, download at: www.microtronics.at/Toolset)
- Account on Cloud server (see "New registration and logging in" on page 115)

4.3 Step-by-step instructions

1. If you do not already have an account on the Cloud server, you can create a free account via the Microtronics Engineering GmbH(cloud.microtronics.at) homepage. Detailed information on creating an account on the Cloud server is provided in chapter "New registration and logging in" on page 115.

2. Install the rapidM2M Toolset . It can be downloaded free of charge from the following website: www.microtronics.at/Toolset. Detailed information on installing the rapidM2M Toolset is provided in chapter "Installation" on page 139.

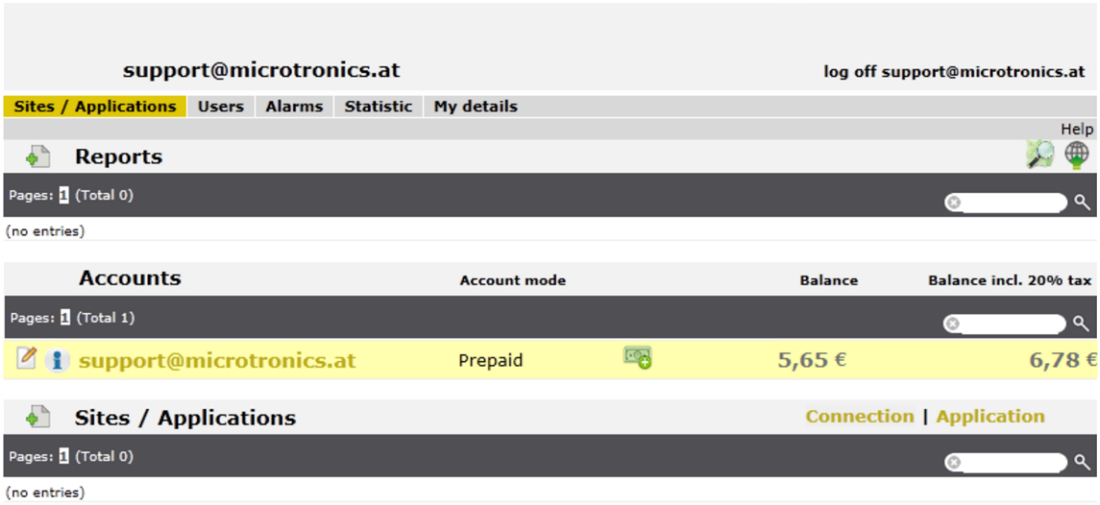
3. Log into the Cloud server (cloud.microtronics.at). Your user name is the e-mail address that you specified when creating the account.



Overview of the homepage

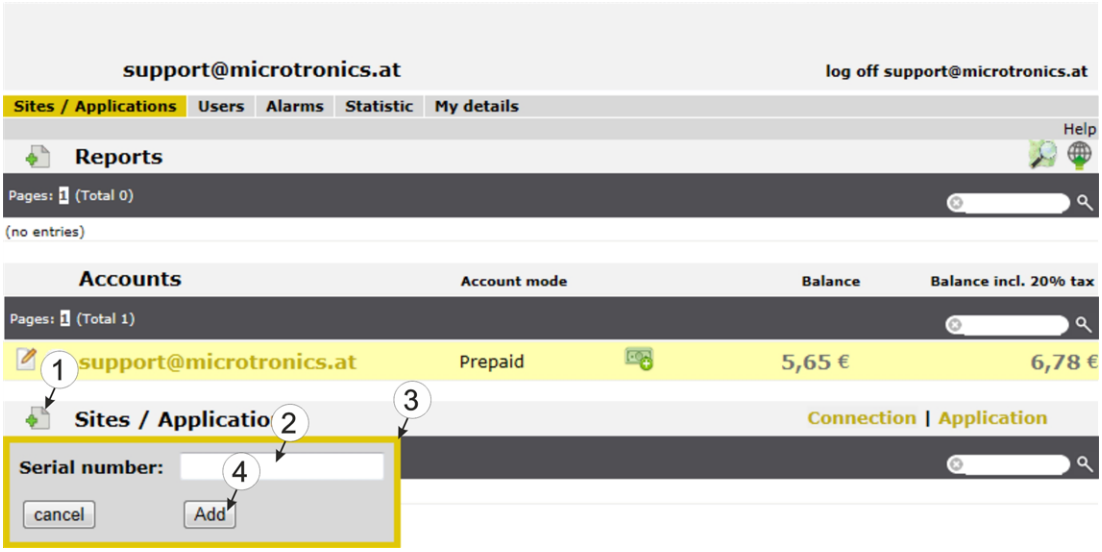
1 Access to the Cloud server	

4. You are now in the "Site/applications" area of the Cloud server (see ""Sites" area" on page 118).



"Sites/applications" area where all of the available sites are displayed.

5. Create a site (see "Creating a new site" on page 123 for details).



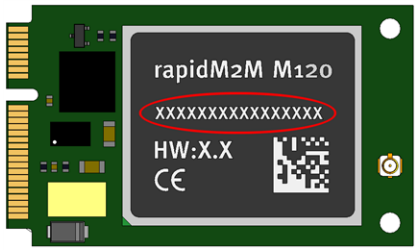
Creating the site

1	"Add new site/application" symbol	3	Input window for creating a new site
2	Field for entering the serial number	4	"Add" button

The serial number is printed on the packaging of the rapidM2M developer package M120 and on the rapidM2M module M120 itself.

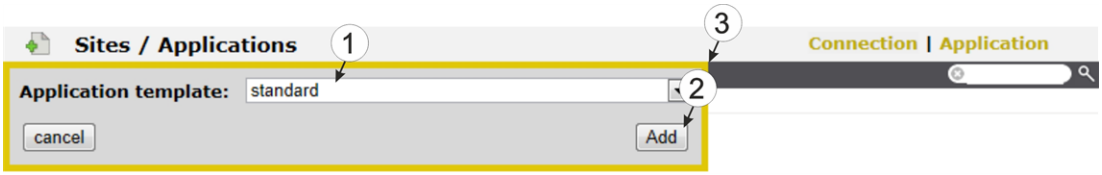


Packaging of the rapidM2M developer package M120



rapidM2M module M120

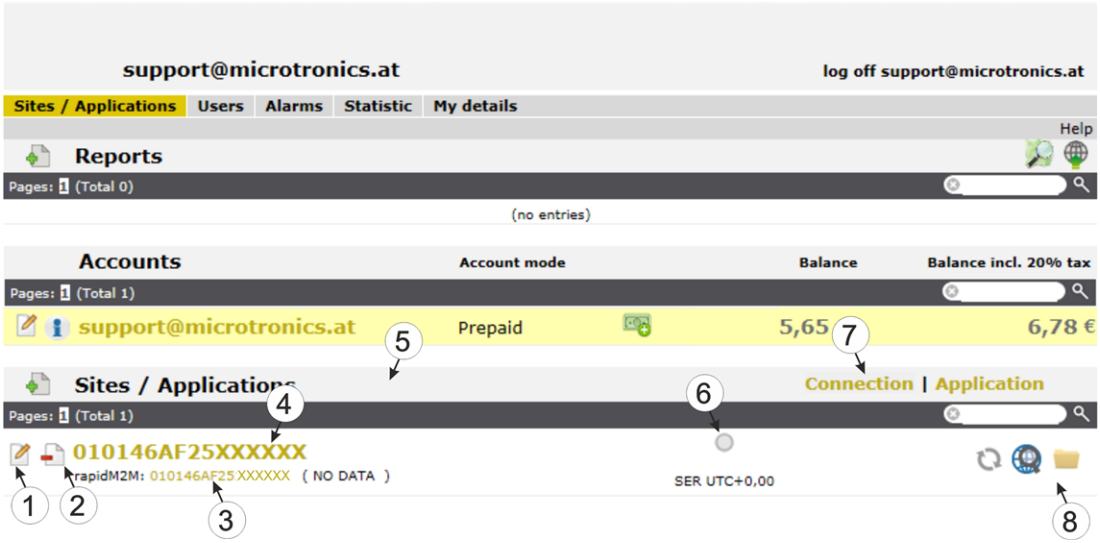
Select "Standard" for the "Application template".



Selecting the application template

1	Dropdown list of available application templates	3	Input window for selecting the application template
2	"Add" button		

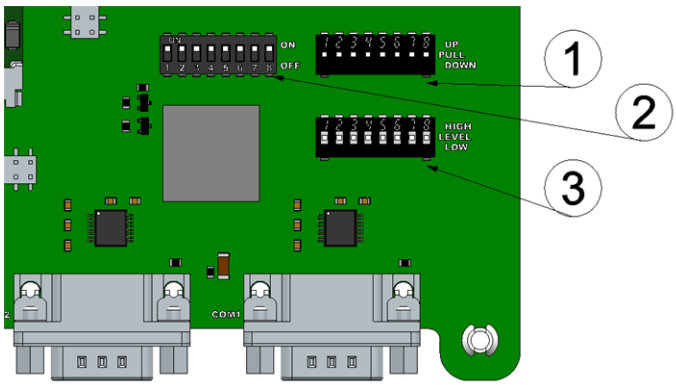
6. The newly added site is now displayed in the list of sites.



"Sites/applications" area where all of the available sites are displayed.

1	Opens the input screen for configuring the site (see "Site configuration" on page 189)
2	Deletes the site/application.
3	Serial number of the device linked to the site. The input screen for configuring the device is opened by clicking on the serial number (see "Device configuration" on page 191).
4	Name of the site/application. The input screen for configuring the site/application is opened by clicking on the name (see "Site configuration" on page 189).
5	List of sites / applications
6	Status of the measurement instrument. In addition to the status symbol, additional symbols are displayed depending on the current operating state. (see "Sites" on page 121 for details)
7	Tabs to switch between the display of the connection-specific and application-specific control elements
8	Connection or application-specific control elements of the site/application (see "Sites" on page 121 for details).

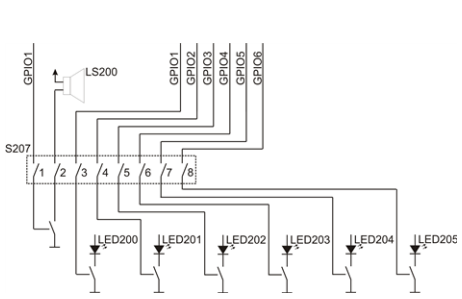
7. To prepare for the script that is used in this tutorial, dip switches S207 - S208 must be set as follows:



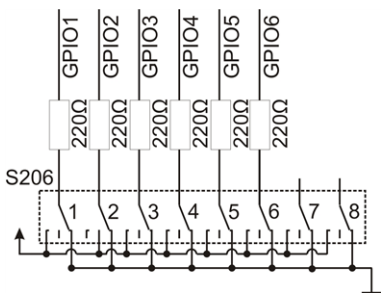
Position of dip switches S207 - S208

1	Dip switch (S206) to activate the pull-up or pull-down resistors for the relevant GPIO (see "Input of digital signals" on page 73)
2	Dip switch (S207) to activate the LEDs for the relevant GPIO and the buzzer (LS200) that is actuated via GPIO1 (see "Output of digital signals" on page 72)
3	Dip switch (S208) to determine whether the relevant GPIO is set to "low" or "high" by pressing the assigned button (see "Input of digital signals" on page 73)

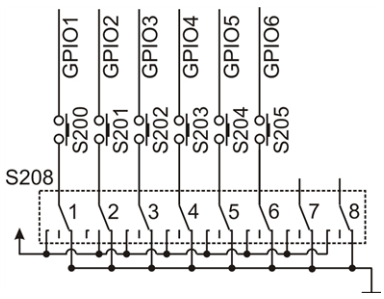
	Dip switch (S207)	Dip switch (S206)	Dip switch (S208)
1	OFF (buzzers deactivated)	o	o
2	OFF (buzzers deactivated)	- (pull down)	+ (pressing button moves GPIO to "high")
3	ON	o	o
4	ON	o	o
5	ON	o	o
6	ON	o	o
7	ON		
8	ON		



Dip switch (S207)

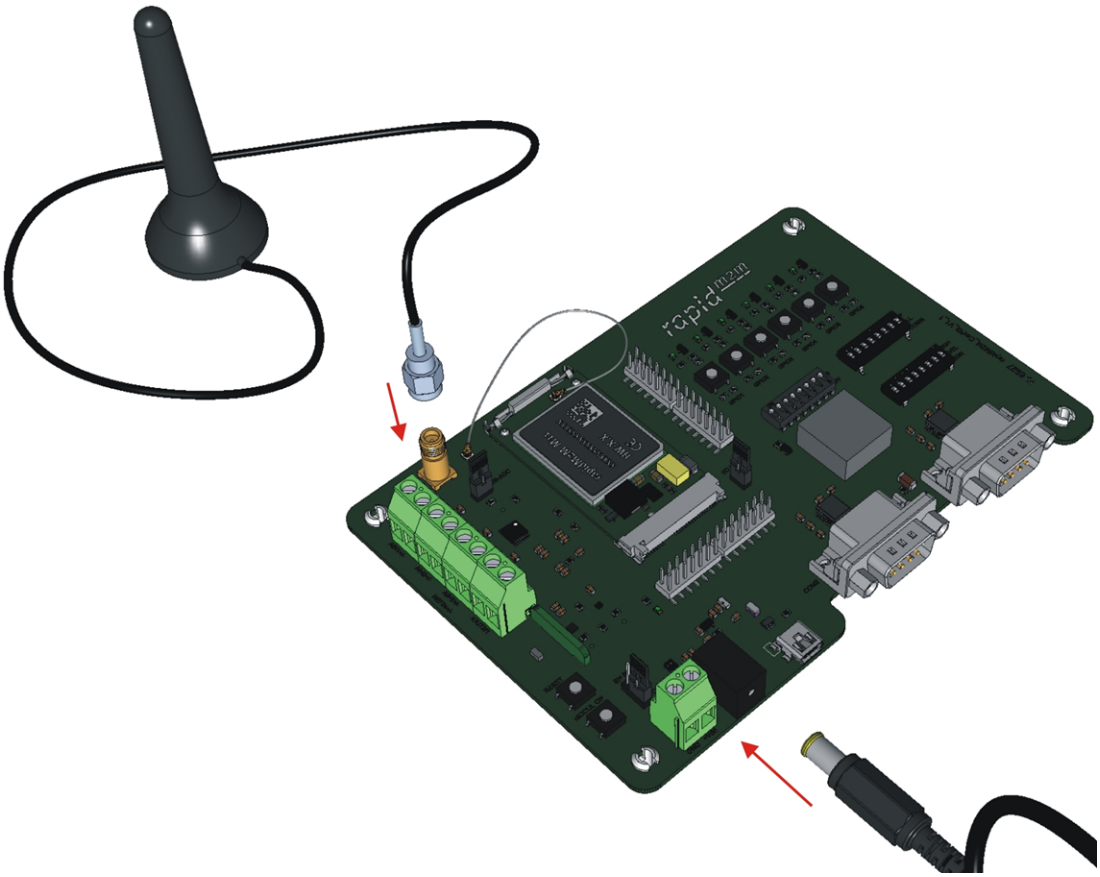


Dip switch (S206)



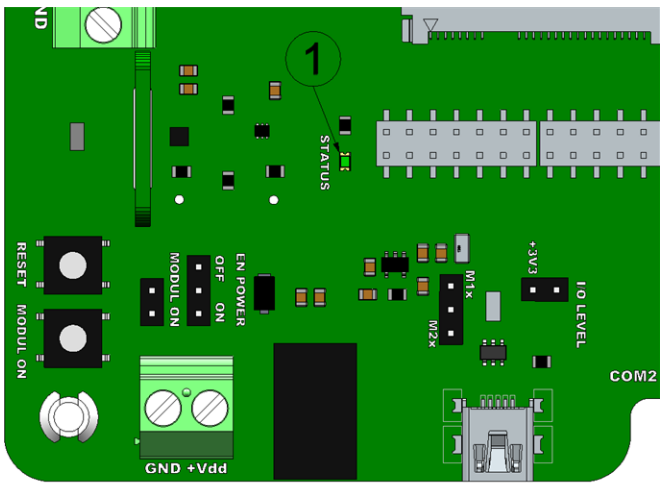
Dip switch (S208)

8. First of all, connect the Magnetic antenna SMA-M 2,5m and then the plug-in power supply to the rapidM2M developer package M120 . In the delivered condition, all of the jumpers are set in such a way that the M120 module starts once the energy supply is established.



Connecting the antenna and power supply

The status LED (LED100) lights up for approx. 1sec. following connection of the energy supply



Position of the status LED (LED100)

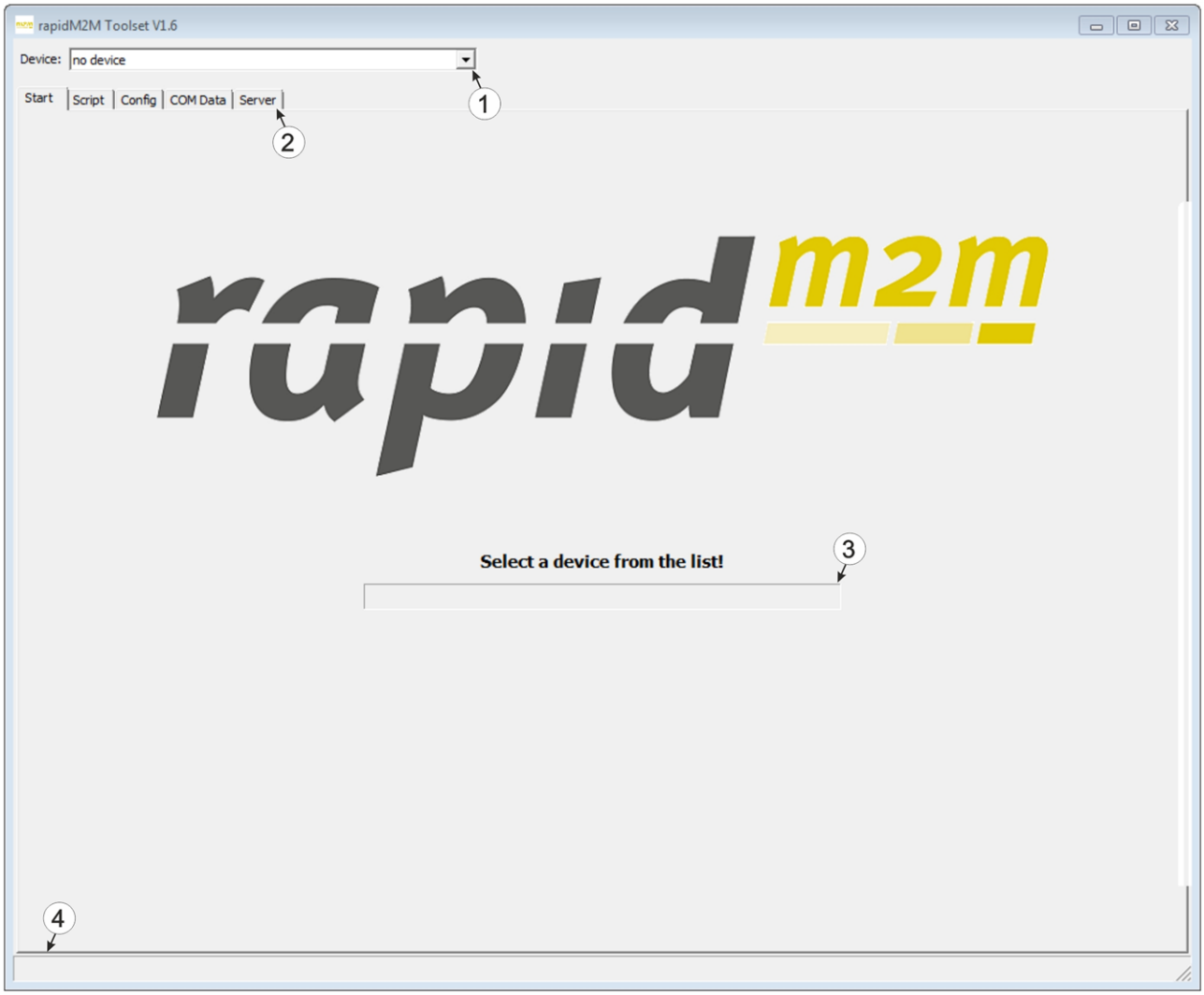
9. In the delivered condition, there is no script in the memory of the M120 module, which means that a connection to the Cloud server is established immediately after the energy supply is connected. Successful communication between the M120 module and Cloud server is indicated by the status symbol.



List of sites / applications

1	Status of the measurement instrument (see "Sites" on page 121 for details)	
		The connection to the Cloud server was successful.
		The M120 module has still not connected to the Cloud server.
2	Indicates that the module is connected to the server and is transferring data. This symbol is hidden again once all of the data has been transferred.	
3	Time of the last connection. A click opens the connection list.	
4	The connection was established via GSM/GPRS. The symbol indicates the signal strength. A click on the symbol displays information about the radio cell used for the last connection and the radio cells taken into account during the last positioning.	

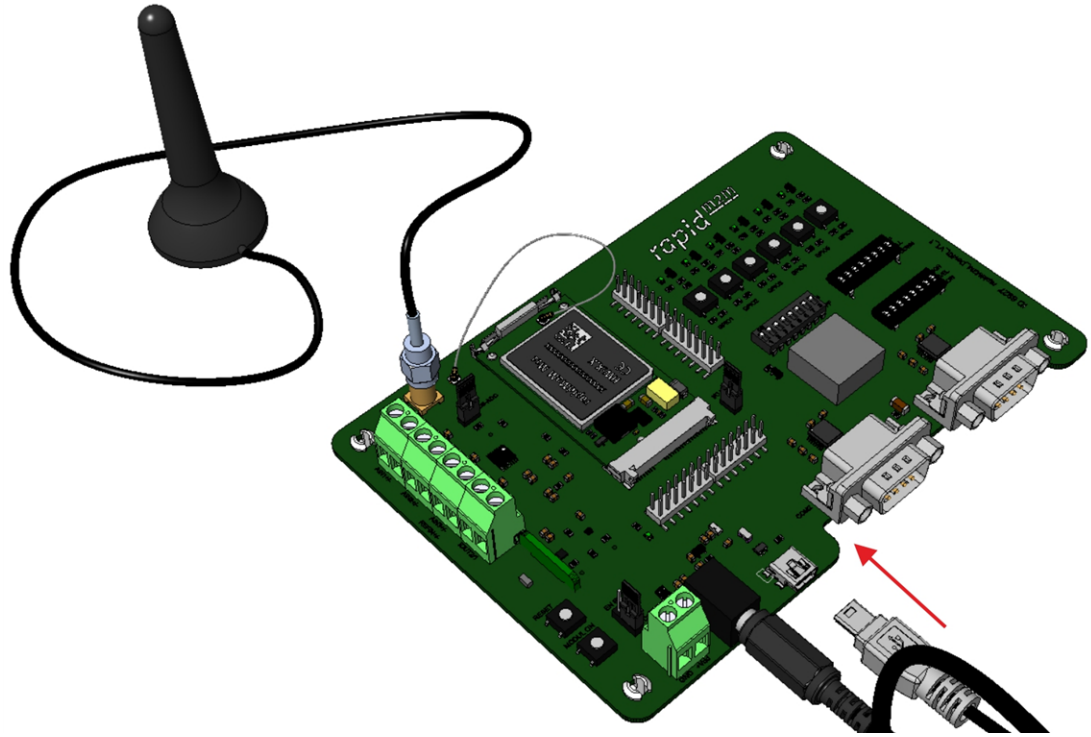
10. Start the rapidM2M Toolset .



"Start" tab (no M120 module selected)

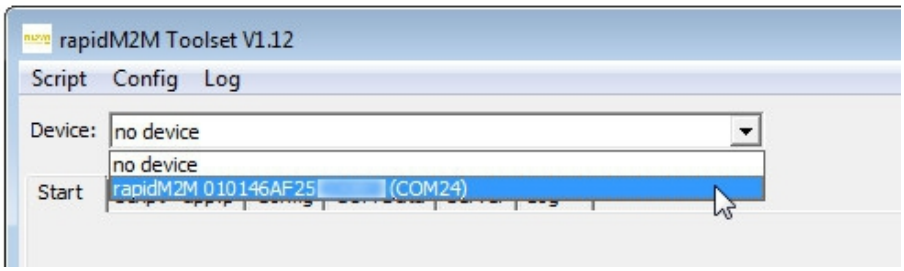
1 Selection of the M120 module	3 Progress bar for establishing the USB connection to the M120 module
2 Selection of the tab for the various functions of the rapidM2M Toolset that are even available without the connected module	4 Indication of the communication status on the USB interface to the M120 module

11. Connect the rapidM2M developer package M120 to the USB interface on your PC.



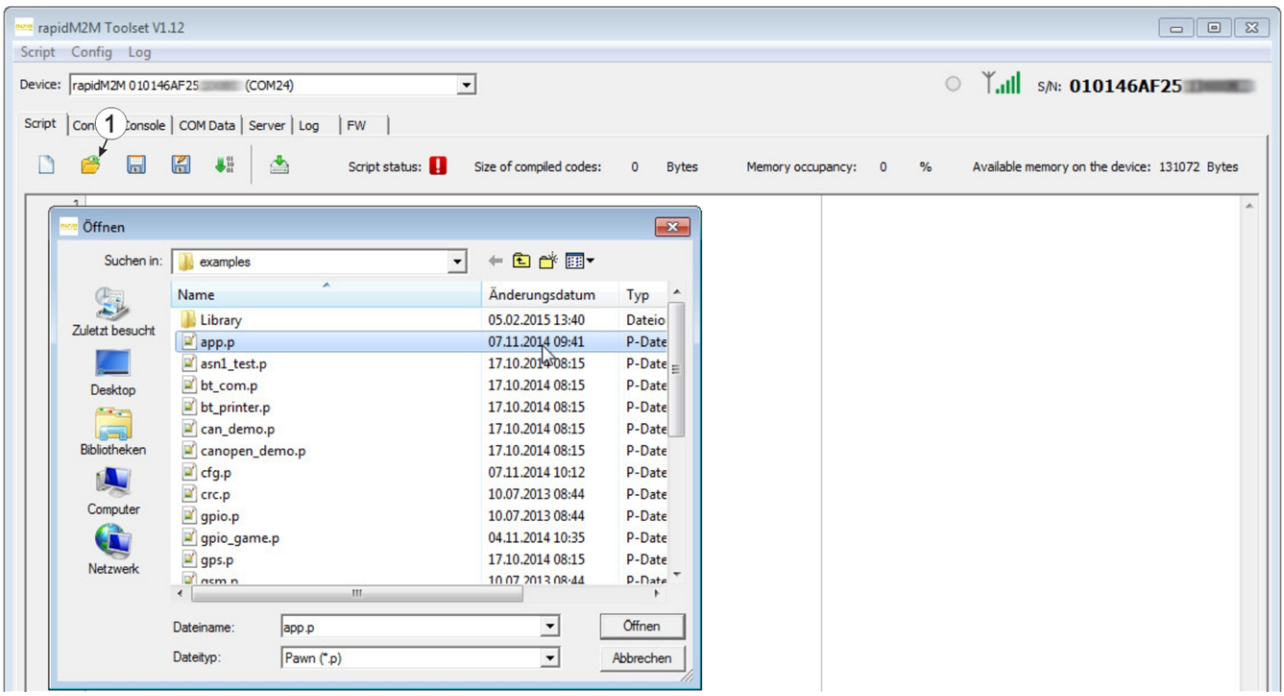
Connecting the USB cable

12. Select the M120 module.



Module selection

13. Open the "app.p" script supplied with the rapidM2M Toolset . It is included in the "examples" sub-folder of the rapidM2M Toolset installation directoryrapidM2M Toolset .

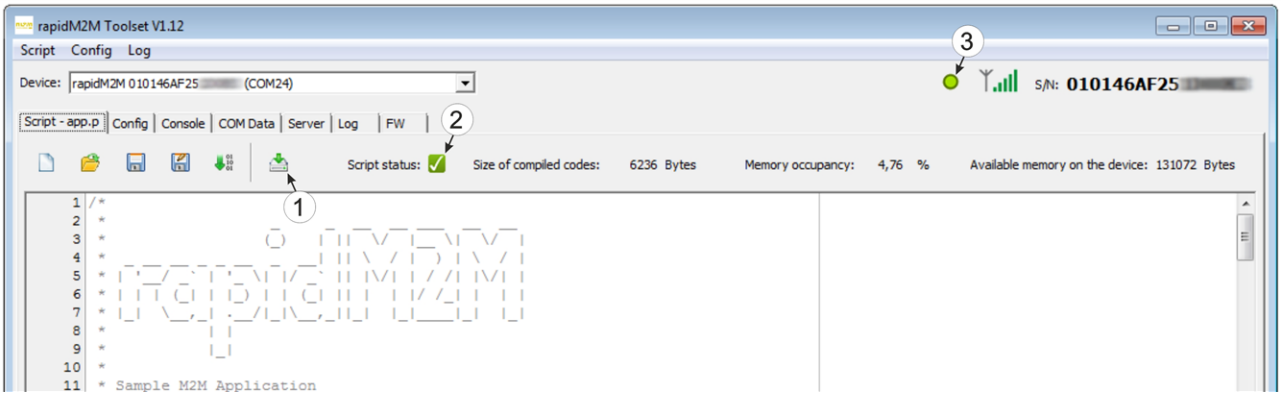


Dialogue to open a saved script.

1 Opens a saved script	

Detailed explanation of the "app.p" script is provided in chapter "Explanation of the rapidM2M specific code blocks of the "app.p" script" on page 45.

14. Transfer the opened script into the M120 module (symbol). The display of the script status then changes to the "Script was successfully loaded in the M120 module" state (symbol). After briefly flashing (connection is being established), the display of the GSM connection status changes to green (connected) as the connection type in the "app.p" script is set to "online" as standard.



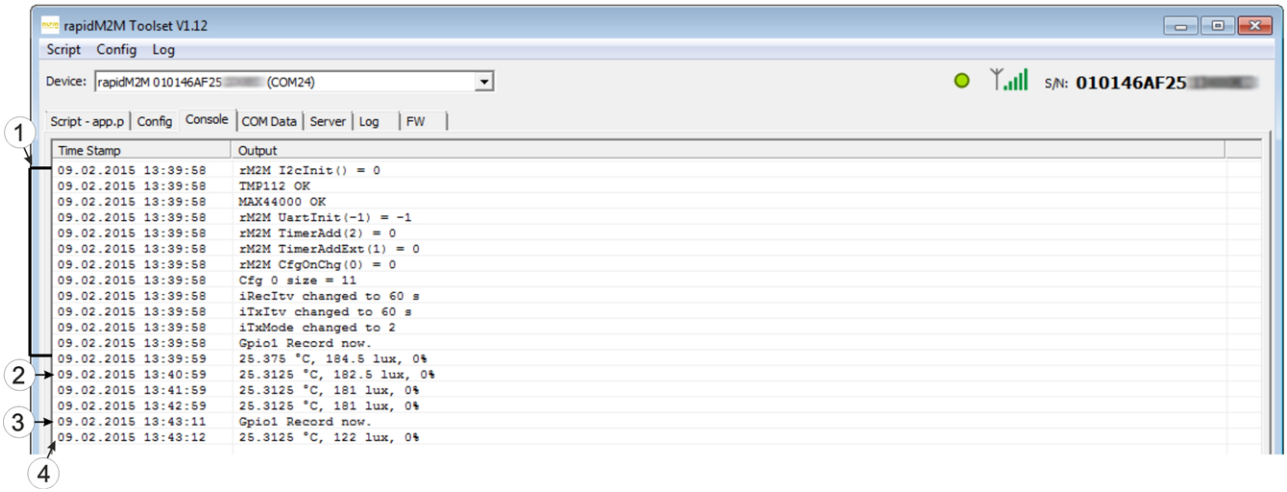
Transferring the script into the M120 module

1	Transfers the opened script into the module. During this process, the script is initially saved and then compiled. The compiled PAWN binary (*.amx) is saved in the same folder as the script.
2	Status of the script transfer(see Script status)
3	Display of the GSM signal strength (see GSM signal strength)

The following step is not mandatory.

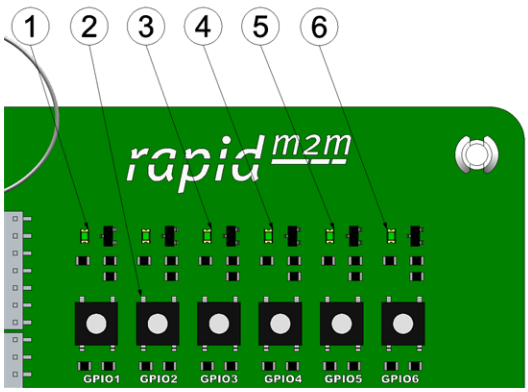
15. If you switch to the "Console" tab, you can view the debug information that is issued by the "app.p" script. This includes the display of measurement values (temperature, ambient light and distance) that were read from both of the sensors and were then transferred to the Cloud server. The "app.p" script generates a new measurement value every 60 seconds as standard. You can also initiate an immediate measurement by pressing the S201 button that is connected to GPIO2. To see a significant change to the measurement value, cover the ambient light and IR proximity sensor MAX44000 (U301) or touch (not recommended) the temperature sensor TMP112 (U300).

> ***Important note:*** *Please be aware that the sensitive electronic components of the rapidM2M developer package M120 can be damaged if they come in contact with static electricity.*

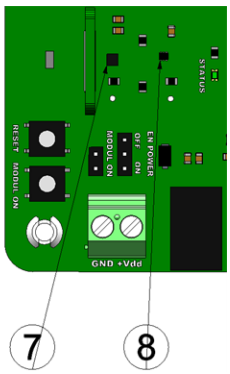


Display of the debug information

1	Initialisation
2	Measurement values generated due to the 60 sec. interval expiring
3	Measurement values were generated by pressing button S201.
4	Measurement values generated by pressing the button (recognisable as the time stamp is directly after the message indicating that the button has been pressed).



Control elements used



Sensors used

1	LED200 (GPIO1): "alive" signal (flashes every second)
2	S201 button (GPIO2): Initiate measurement immediately
3	LED202 (GPIO3): can be activated/deactivated via configuration parameter
4	LED203 (GPIO4): can be activated/deactivated via configuration parameter
5	LED204 (GPIO5): can be activated/deactivated via configuration parameter
6	LED205 (GPIO6): can be activated/deactivated via configuration parameter
7	Temperature sensor TMP112 (U300): "Temperature [°C]" measurement value
8	Ambient light and IR proximity sensor MAX44000 (U301): "Ambient light [lux]" and "Distance [%]" measurement values

16. The measurement values are now already recorded and are immediately transferred to Cloud server due to the "online" connection type. The values are currently still saved in the raw measurement data area ("rm2mraw"). This is why "NO DATA" is still shown in the display for the available data period. This display relates to measurement data channel 0 ("histdata0"). To be able to use the measurement data generated by the "app.p" script and the provided configuration via the interface of the Cloud server, the structure of the measurement data and configuration must be known by the server. The "myDatanet Connector " must be defined for this purpose. A description is provided in the next steps.



List of sites / applications

1 Display of the available data period	2 Indicates that the module is connected to the server and is transferring data.

17. Open the input screen to configure the site/application by clicking on the name of the site/application in the list of sites/applications.

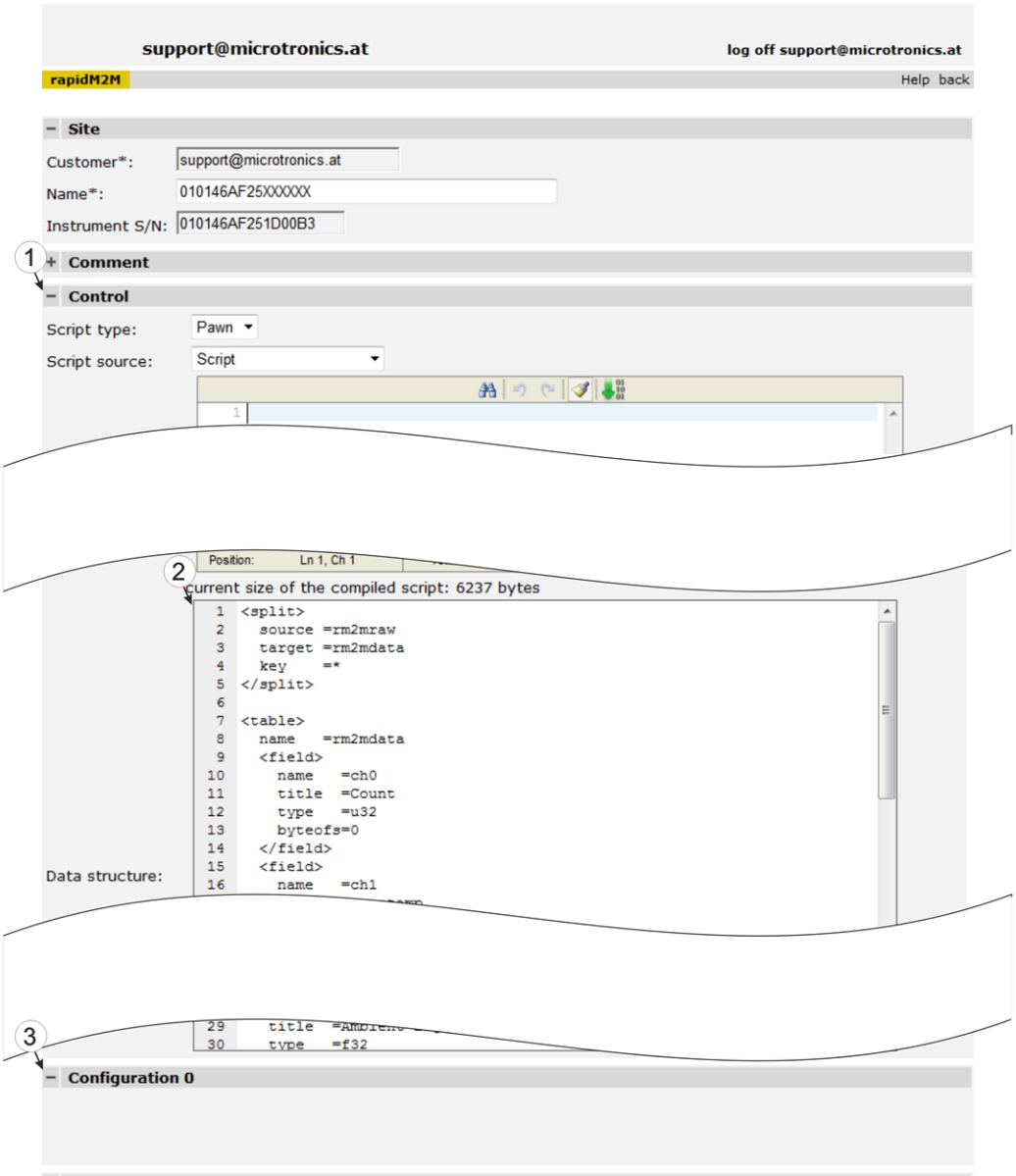


List of sites / applications

1 Name of the site/application. The input screen for configuring the site/application is opened by clicking on the name (see "Site configuration" on page 189).

18. Open the "Control" configuration section by clicking on the "+" symbol next to the "Control" inscription. Copy the content of the "app_connector.txt" file supplied with the rapidM2M Toolset that is included in the "examples" sub-folder of the rapidM2M Toolset installation directoryrapidM2M Toolset rapidM2M Toolset into the input screen for configuring the myDatanet Connector . Then click on the "Save" button located at the very bottom of the input screen for configuring the site/application. This closes the current input screen and you are returned to the list of sites/applications.

    A detailed explanation is provided in chapter "Explanation of the connector definition included in the "app_connector.txt" file" on page 50.



Input screen for configuring the site/application

1	Configuration section for selecting the script type, script source and specifying the data structure (myDatanet Connector). Detailed information is provided in chapter "Control" on page 189.
2	Input window for configuring the myDatanet Connector (see "myDatanet Connector " on page 277)
3	Configuration section to display and change the parameters of configuration memory block 0. This section is empty as long as the Cloud server does not know the structure of the configuration memory block (myDatanet Connector not defined).

19. The available data period can now be displayed correctly as the data is copied using the split tag upon receipt in measurement data channel 0 ("histdata0") (see "Split tag" on page 278) and the Cloud server knows the structure of the measurement data generated by the "app.p" script (see "Data structure" on page 280).

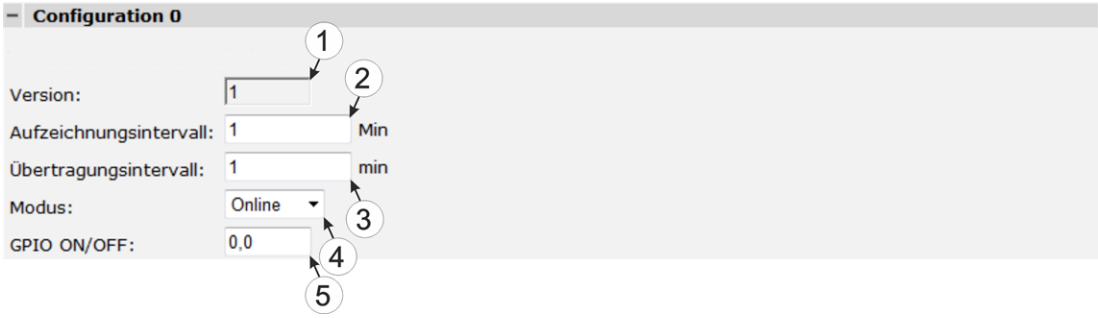


List of sites / applications

1	Display of the available data period

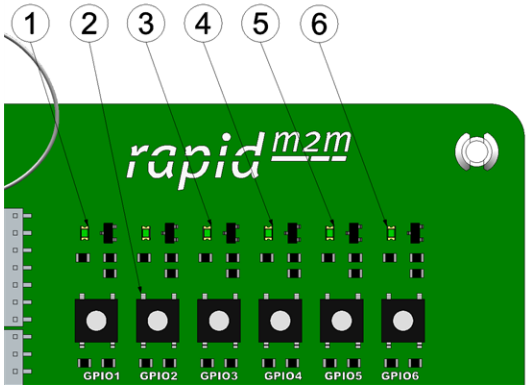
The following step is not mandatory.

20. Open the input screen to re-configure the site/application and test the effects of changes to the configuration.



Input screen for configuring the site/application, "Configuration 0" section

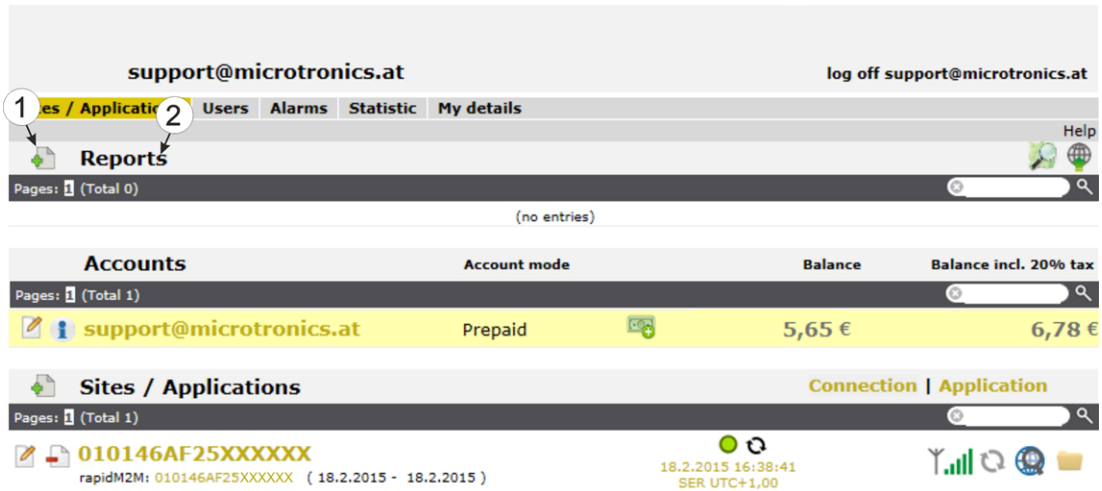
1	Display of the current version of the configuration (read only). Is specified by the script and should be changed if the structure of the configuration changes (e.g. addition of another parameter).	
2	Time between measurement data recordings	
3	Time between transmissions	
4	Interval	The device registers in the transmission cycle.
	Wakeup	The device connects to the server in the transfer interval and can be placed into the ALOHA transmission mode via the server. 
	Online	The device does not separate the connection and continuously transmits the measurement data.
5	Enables the selective activation/deactivation of the LEDs. Bit0: LED202 (GPIO3); bit set -> LED ON Bit1: LED203 (GPIO4); bit set -> LED ON Bit2: LED204 (GPIO5); bit set -> LED ON Bit3: LED205 (GPIO6); bit set -> LED ON LED205 (GPIO6) lights up if this parameter is set to "8".	



Control elements used

1	LED200 (GPIO1): "alive" signal (flashes every second)
2	S201 button (GPIO2): Initiate measurement immediately
3	LED202 (GPIO3): can be activated/deactivated via configuration parameter
4	LED203 (GPIO4): can be activated/deactivated via configuration parameter
5	LED204 (GPIO5): can be activated/deactivated via configuration parameter
6	LED205 (GPIO6): can be activated/deactivated via configuration parameter

21. Create a report to display the measurement values (see "Reports" on page 119).



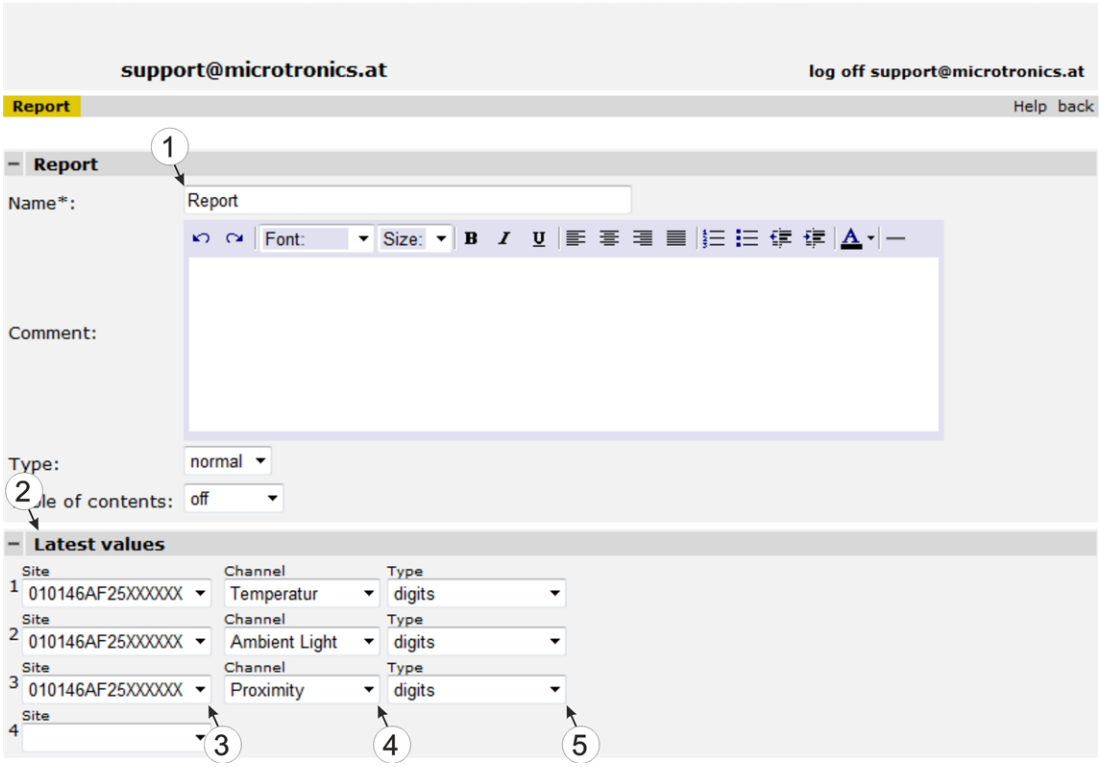
"Sites/applications" area where all of the available sites are displayed.

1	Adds a new report	2	List of reports

22. Configure the "Last measurement values" display element as follows. This ensures that the last measurement values of the relevant measurement channels are displayed next to the name of the report in the list of reports. By clicking on the "Save" button, you will return to the "Sites/applications" area in which all of the available sites are displayed.

    Detailed instructions on reports are provided in the myDatanet Server Manual (805002).

	Site	Channel	Type
1	{Serial number}	Temperature	Digits
2	{Serial number}	Ambient light	Digits
3	{Serial number}	Proximity	Digits



Input screen for the basic configuration of the report

1	Freely selectable name for the report
2	Configuration of the "Last measurement values" display elements that are shown next to the name of the report in the list of reports (see "Reports" on page 119).
3	Selection of the site. The element can obtain its data from different sites. All of the current customer's sites are available to be selected.
4	Site channel that should be displayed
5	Selection of the type of display element to be used

The following step is not mandatory.

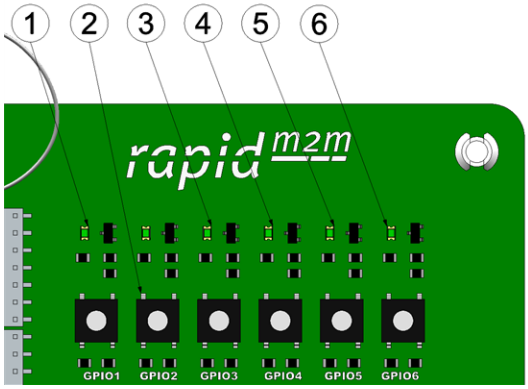
23. The current measurement values are now displayed directly next to the name of the report. The measurement values are updated every 60 sec. provided you have not changed the standard configuration (record interval: 1 min; mode: online). You can also initiate an immediate measurement by pressing the S201 button that is connected to GPIO2. To see a significant change to the measurement value, cover the ambient light and IR proximity sensor MAX44000 (U301) or touch (not recommended) the temperature sensor TMP112 (U300).

    Important note: *Please be aware that the sensitive electronic components of the rapidM2M developer package M120 can be damaged if they come in contact with static electricity.*

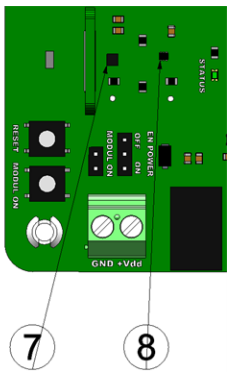


"Sites/applications" area where all of the available sites are displayed.

1	Opens the input screen for the basic configuration of the report
2	Clicking on the name of the report will open the input screen to configure and select the elements.
3	Display elements for the last measurement value of a measurement channel (max. four per report). The site, channel and type of display element is selected in the basic configuration of the report (see "Basic configuration" on page 119).



Control elements used



Sensors used

1	LED200 (GPIO1): "alive" signal (flashes every second)
2	S201 button (GPIO2): Initiate measurement immediately
3	LED202 (GPIO3): can be activated/deactivated via configuration parameter
4	LED203 (GPIO4): can be activated/deactivated via configuration parameter
5	LED204 (GPIO5): can be activated/deactivated via configuration parameter
6	LED205 (GPIO6): can be activated/deactivated via configuration parameter
7	Temperature sensor TMP112 (U300): "Temperature [°C]" measurement value
8	Ambient light and IR proximity sensor MAX44000 (U301): "Ambient light [lux]" and "Distance [%]" measurement values

4.4 Explanation of the rapidM2M specific code blocks of the "app.p" script

4.4.1 Initialising the sensors

Initialises the I²C interface of the M120 module and temperature sensor TMP112 (U300), as well as the ambient light and IR proximity sensor MAX44000 (U301) located on the rapidM2M board for developers . Debug information""Console" tab" on page 147rapidM2M Toolset is also printed on the standard output via the "print" and "printf" functions (see "Console functions" on page 197) and can then be evaluated via the "Console" tab (see ""Console" tab" on page 147) of the rapidM2M Toolset .

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

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

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

4.4.2 Initialising the timers

Initialises the 1 sec. timer that is used to generate the record and transmission interval and determines the callback function called up when the timer elapses. The callback function initiates a connection to the Cloud server when the transmission interval elapses. It also completes the recording when the record interval elapses and issues the current measurement values on the standard output via the "printf" function (see "Console functions" on page 197).

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

Initialises a 100 ms timer with auto reload. This is used to read the values from the (TMP112 (U300) and MAX44000 (U301)) sensors. The timer is also used to monitor whether button S201 (GPIO2) was pressed to trigger an immediate recording. The callback function, that is called up following expiry of the timer, is also determined.

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

4.4.3 Determining the callback for handling the configuration

Specifies the callback function that should be called up if one of the configuration memory blocks has changed.

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

4.4.4 Setting the connection type

Sets the connection type stored in the configuration. "online" mode is saved as standard in the "app.p" script. The connection to the Cloud server is therefore immediately established at this bit of the code.

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

4.4.5 Saving measurement data

The "rM2M_GetTime()" function reads out the current time stamp (seconds since 31.12.1999).

> ***Note:*** *This is only used as another measurement value for the record. It is not used as the time stamp for the data record. The "rM2M_RecData()" function generates the time stamp for the data record itself as the first parameter is "0".*

The data area of the measurement data record is created by calling up the "rM2M_SetPacked()" function several times. The definition of the myDatanet Connector is also based on this structure (see "Measurement data channel 0" on page 51). This data area is then transferred to the "rM2M_RecData()" function to be recorded.

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

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

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

4.4.6 Monitoring button S201 to initiate an immediate recording

Cyclically checks the state of button S201 (GPIO2). If it is detected that the button has been pressed, the counter for the remaining time until the next recording is set to 0, which immediately triggers a recording.

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

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

4.4.7 Reading the sensor values

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

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

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

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

4.4.8 Reading the configuration

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

Firstly, it is checked if at least 1 byte could be read when reading the configuration block via the "rM2M_CfgRead()" function. The first byte of the "app.p" script contains the version number of the configuration. If the version number has changed (e.g. due to another parameter) or the configuration is not the expected size, a default configuration must be generated and saved. The data area of the configuration block is created by calling up the "rM2M_SetPacked()" function several times. The definition of the myDatanet Connector is also based on this structure (see "Configuration memory block 0" on page 52). This data area is then transferred to the "rM2M_RecData()" function to be stored.

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

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

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

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

Activates or deactivates the LED202 - LED205 (GPIO3 - GPIO5) in accordance with the configuration entered via the interface of the Cloud server.

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

4.5 Explanation of the connector definition included in the "app_connector.txt" file

4.5.1 Split tag

Code	Explanation

`<split>`
`  source =rm2mraw`
`  target =rm2mdata`
`  key    =*`
`</split>`
	Start of the split tag
	The data from the raw measurement data area should be copied.
	The data is copied to measurement data channel o[1].
	"*" means that all of the data records are copied from the source to the target.
	End of the split tag

[1] "rm2mdata" is equal to "histdatao". Although "rm2mdata" will continue to be supported for the purpose of downward compatibility, "histdatao" should be used.

Detailed information is provided in chapter "Split tag" on page 278.

4.5.2 Measurement data channel 0

The data area of the measurement data record is structured as follows:

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

The myDatanet Connector must therefore be defined as follows:

Code	Explanation
`<table>`	Start of the table tag
`name =rm2mdata`	The following description applies to measurement data channel 0[1].
`<field>`	Start of the field definition for a "Count" measurement value
`name =ch0`	A description of the attributes is provided in
`title =Count`	"Attributes of the field definition" on page 281.
`type =u32`	
`byteofs=0`	
`</field>`	
`<field>`	Start of the field definition for the "Time stamp" measurement value
`name =ch1`	
`title =Time stamp`	
`type =u32`	
`byteofs=4`	
`</field>`	
`<field>`	Start of the field definition for the "Temperature" measurement value
`name =ch2`	
`title =Temperature`	
`type =f32`	
`byteofs=8`	
`</field>`	
`<field>`	Start of the field definition for the "Ambient light" measurement value
`name =ch3`	
`title =Ambient light`	
`type =f32`	
`byteofs=12`	
`</field>`	
`<field>`	Start of the field definition for the "Proximity" measurement value
`name =ch4`	
`title =Proximity`	
`type =u32`	
`byteofs=16`	
`</field>`	
`</table>`	

[1] "rm2mdata" is equal to "histdatao". Although "rm2mdata" will continue to be supported for the purpose of downward compatibility, "histdatao" should be used.

Detailed information is provided in chapter "Data structure" on page 280.

4.5.3 Configuration memory block 0

The data area of the configuration block is structured as follows:

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

The myDatanet Connector must be defined as follows:

Code	Explanation

`<table>`
`  name  =rm2mconfig0`
`  <field>`
`  name  =field0`
`  title  =Version`
`  decpl  =0`
`  type  =u8`
`  units  =`
`  byteofs=0`
`  edit  =8`
`  </field>`
	Start of the table tag
	The following description applies to configuration 0[1)].
	Start of the field definition for the "Version" parameter
	A description of the attributes is provided in
	"Attributes of the field definition" on page 281.

`  <field>`
`  name  =field1`
`  title  =Record interval`
`  units  =Min`
`  decpl  =0`
`  vscale =0.01666666667`
`  type  =u32`
`  byteofs=1`
`  </field>`
	Start of the field definition for the
	"Record interval" measurement value

`  <field>`
`  name  =field2`
`  title  =Transmission cycle`
`  units  =min`
`  decpl  =0`
`  vscale =0.01666666667`
`  type  =u32`
`  byteofs=5`
`  </field>`
	Start of the field definition for the
	"Transmission cycle" measurement value

`  <field>`
`  name  =field3`
`  title  =Mode`
`  editmask=0=Interval;1=Wakeup;2=Online`
`  type  =u8`
`  byteofs=9`
`  </field>`
	Start of the field definition for the "Mode" measurement value

`  <field>`
`  name  =field4`
`  title  =GPIO ON/OFF`
`  type  =u8`
`  byteofs=10`
`  </field>`
`</table>`
	Start of the field definition for the "Proximity" measurement value

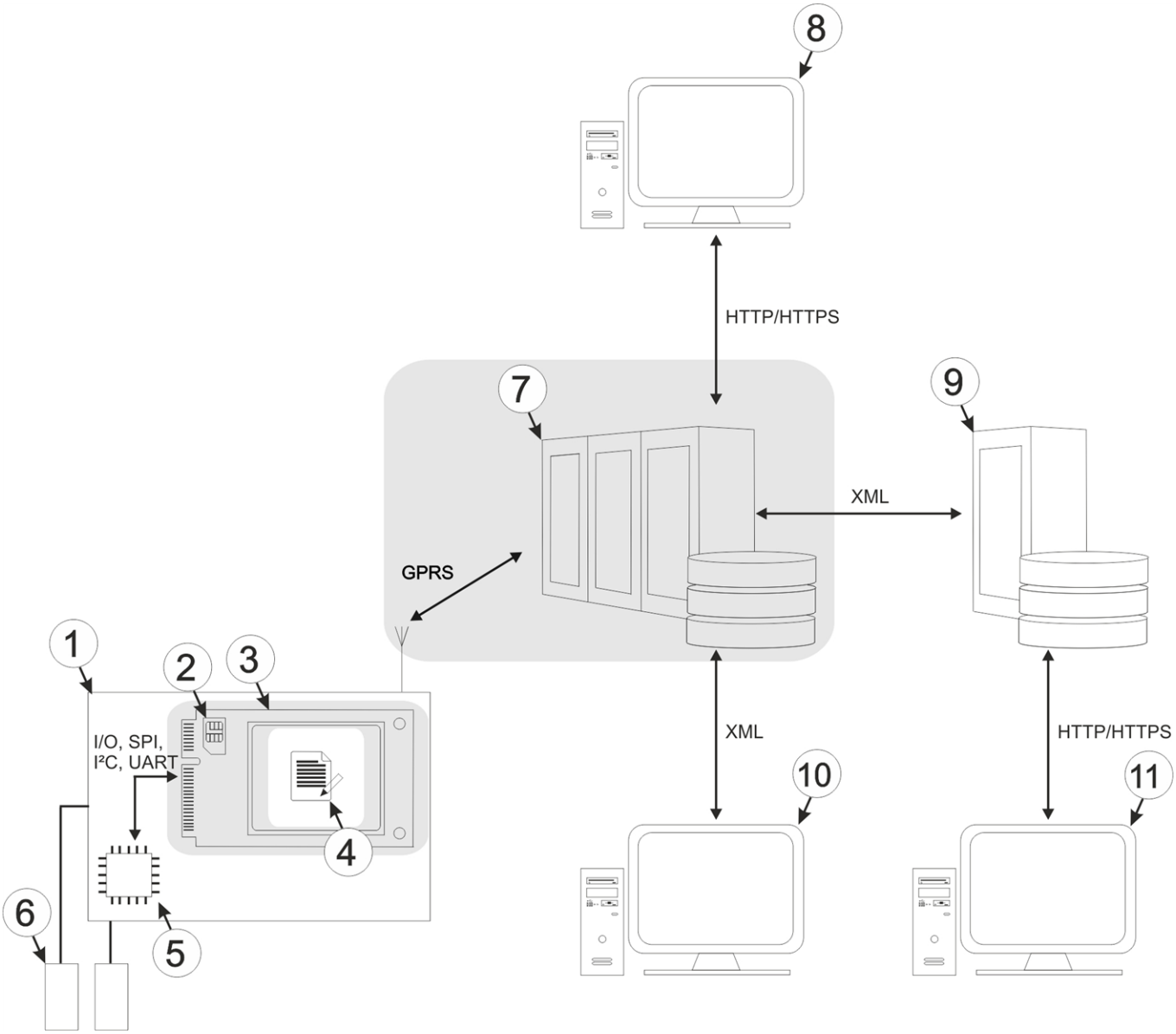
[1] "rm2mconfigo" is equal to "configo". Although "rm2mconfigo" will continue to be supported for the purpose of downward compatibility, "configo" should be used.

Detailed information is provided in chapter "Data structure" on page 280.

6872
6872
6872

Chapter 5 Functional principle

The rapidM2M module is based on the basic idea of "storage-2-storage" data transmission. For this type of data transmission, the M120 module and the Cloud server must not know about the logical content of the data blocks. Therefore, the only aim is to transfer a block of data from A to B. In the graphic below, all of the components that are part of the rapidM2M are illustrated in grey. All of the components that must be provided/created by the customer are illustrated in black.



Functional principle of the rapidM2M

1	Device (customer-specific hardware) including installed M120 module
2	Managed Service SIM card including data transmission
3	M120 module (see "rapidM2M module M120 " on page 97)
4	Application created by the customer (pawn script) that handles the exchange of data with the electronic measuring equipment and the recording of data (see "Pawn script" on page 193)
5	Customer-specific electronic measuring equipment
6	Sensor that is connected to the device (customer-specific hardware) (optional)
7	Cloud server to which the data is transferred.
8	Client that accesses the interface of the Cloud server via the web browser (see "Cloud server" on page 115). The myDatanet Connector must be configured correctly for this purpose (see "myDatanet Connector " on page 277).
9	Customer-specific server that provides clients with their own interface. The customer-specific server obtains the data via the XML interface of the Cloud server (see "XML interface" on page 153).
10	Client, on which a PC program is running, that obtains its data via the XML interface of the Cloud server (see "XML interface" on page 153).
11	Client that accesses the interface of the customer-specific server via the web browser

Functions and components provided by rapidM2M :

- M120 module:

  Programmable (see "Pawn script" on page 193) communication module with integrated memory and standardised microcontroller interfaces (see "Serial interfaces" on page 108)

- Managed Service

  Managed Service is the basis for operating your devices and provides you with 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.

- Cloud server

  Database for saving the measurement data and configurations. Data is either accessed via the XML interface of the server (see "XML interface" on page 153) or web interface of the server. The Cloud server must be notified of the structure of the data blocks if the server is to be accessed via the web interface. This is completed by means of the myDatanet Connector (see "myDatanet Connector " on page 277).

Functions and components provided by the customer:

- Electronic measuring equipment:

  Electronic system for connecting the M120 module and recording the measurement values. This system can either comprise its own measurement controller that transfers its data via an interface to the M120 module or just the sensor electronics. It is possible to program the entire measurement application into the M120 module so that no additional measurement controller is required.

- Application (pawn script):

  The firmware of the M120 module only manages the synchronisation of the measurement data and configurations between the M120 module and Cloud 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 M120 module (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

- Customer-specific server with web interface for the clients:

  It is therefore possible to create an individual web interface for the clients. Using this method, the data is read out of the Cloud server via the XML interface (see "XML interface" on page 153) 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 myDatanet Connector (see "myDatanet Connector " on page 277) when developing a M2M/IoT application. It specifies the various data structures (measurement data, configurations, etc.) that are valid for the script as well as the Cloud server. The definitions of the myDatanet Connector also apply for accessing the data of the Cloud server via the XML interface .

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 Automatic selection of the GSM network

The GSM network to which the device should register must be selected, as the rapidM2M developer package M120 is equipped with a SIM chip that provides a mobile connection via a variety of international service providers (see www.microtronics.at/footprint). This is completed automatically by the device. During this process it is taken into consideration if a connection to a certain GSM network is possible although the use of this GSM network repeatedly causes transmission errors. Such a GSM network is added to a ban list and is ignored during the next login attempt.

The following conditions ensure that the rapidM2M developer package M120 logs itself into a GSM network:

- PowerOn (only if no script is present or the script has been deactivated due to an error)
- Initiation of the transmission if the "Interval" connection type was selected (see "rM2M_ TxSetMode()")
- Activation of the "Interval & Wakeup" or "Online" connection type (see "rM2M_ TxSetMode()")

1. The rapidM2M developer package M120 checks whether any errors occurred during the last login attempt.

   No errors:

   The rapidM2M developer package M120 logs into the same GSM network that was also used during the last attempt.

   Error:

   Continue with step 2

   > ***Note:*** *A transmission error is not the same as an error that occurs when logging into the GSM network. It is possible for the login to the GSM network to work correctly but that a transmission error still occurs.*

The following steps are only completed if an error was detected during step 1:

2. The device searches for GSM networks within its receiving range and creates a list of available GSM networks.
3. A search of the list of available GSM networks is completed and the strongest GSM network is selected that is not on the "ban list" and is assigned to one of the service providers supported by the integrated SIM chip (see www.microtronics.at/footprint).

   > ***Note:*** *An explanation detailing the conditions when a GSM network is added to the "ban list" is provided in "Ban list" on page 59.*

5.2.1 Ban list

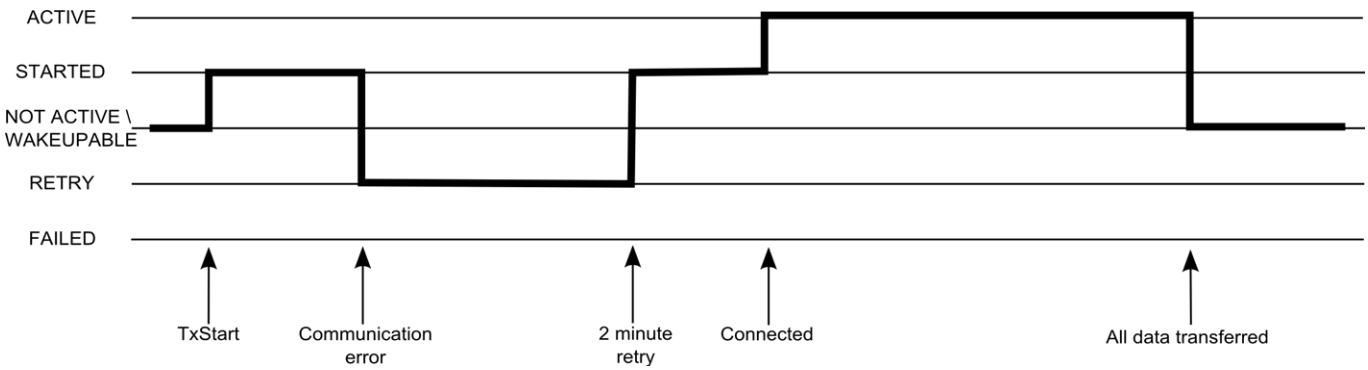
The ban list provides space for 5 entries and is not saved in a volatile memory. This means that it is also still available following a PowerOn.

If there is a GPRS error (see error codes -975 to -993 in chapter "Modem error" on page 292), then an internal counter is increased by 2 . If a transmission was completed without errors, i.e. all of the data was transferred, this internal counter is reduced by 1 . The current GSM network is added to the ban list if the internal counter reaches the value 6 . The relevant GSM network is automatically added to the ban list, if the "MODEM CMME ERROR" error occurs (see "Modem error" on page 292) with parameter 32 (NETWORK NOT ALLOWED) or 529 (MODEM CMME ERROR). The ban list is deleted if no GSM network can be selected from the list of available GSM networks in the receiving range taking the ban list into consideration. Another search for GSM networks in the receiving range is completed once the ban list is deleted and the network selection is completed with the ban list that is now empty (see step 3 in chapter"Automatic selection of the GSM network" on page 59).

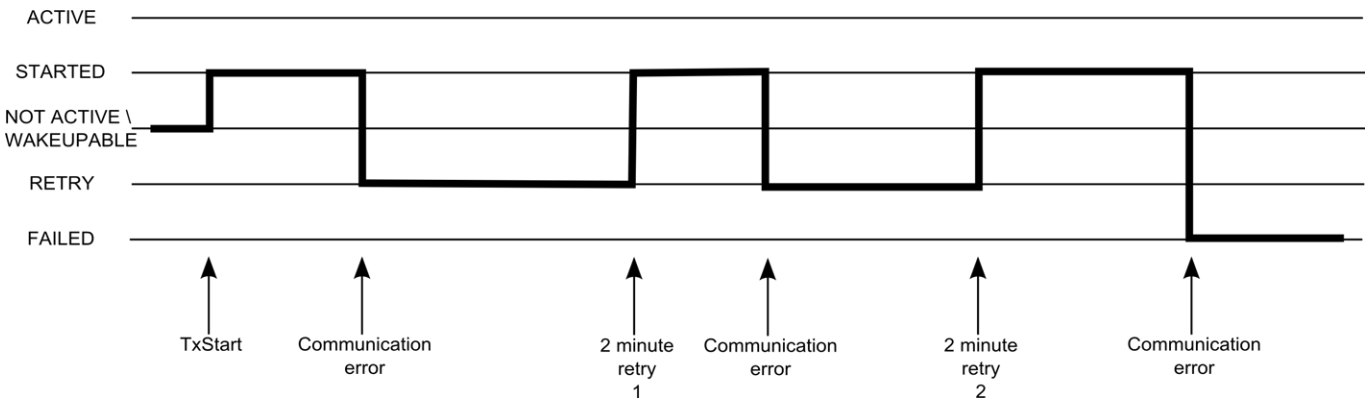
The quality of the data transfer is monitored in addition to the establishment of the GPRS connection. If the data transmission fails 10 times in succession despite an existing GPRS connection, the currently selected GSM network is also added to the ban list. The same applies in this case, which means that if, when taking the ban list into consideration, there is no further GSM network available, the ban list is deleted and selection of the strongest GSM network is restarted.

5.3 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. The connection is attempted up to 2 times.



Connection could be established during first retry.



Connection could not be established despite 2 retries.

ACTIVE
Connection to the Cloud 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 Cloud server (see "Sites" on page 121).

RETRY
The system waits 2min until the next connection attempt.

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.

5.3.1 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.

5.3.2 Connection abort during a script download

The device reacts to a connection abort during script download with the standard retry sequence (2 connection attempts at intervals of 2min). In addition to this, the "SCRIPT_ERR, SCRIPT UPDATE ERROR" log entry is entered in the device log as soon as the device has detected the connection abort. As the existing script 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 script is located is deleted at the start of the script download.

5.4 Error handling

The following transmission mechanisms have been integrated in the firmware to ensure that problems with the script can be diagnosed and resolved remotely. In the event that there is no script, a connection to the Cloud server is established every 24h . This backup interval is set to 1h if an existing script 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 Cloud (see "Sites" on page 121).

5.5 Registration memory blocks

These 4 blocks each with 1kB provide the option of storing device-specific data and synchronising it with the Cloud server. The blocks only differs with regard to their access options. This results in predefined intended uses that are described in the following table:

Memory block	Access	Purpose
System-specific data		
REG_SYS_OTP [1)]	**readable:** Script, Cloud server	System information that is written once as part of the production process
REG_SYS_FLASH [1)]	**readable:** Script, Cloud server	System information that must be able to be changed during operation
Application-specific data		
REG_APP_OTP	**readable:** Script, Cloud server **writeable:** Script	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	**readable:** Script, Cloud server **writeable:** Script	Application-specific information that must be able to be changed during operation

[1)] Writing data in these two memory blocks is reserved for the manufacturer.

> ***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 307).*

The "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()" and "rM2M_RegSetValue()" functions are available for accessing the registration memory blocks.

5.5.1 REG_SYS_FLASH

If the site that is linked to the device was created from an application template (see myDatanet Server Manual 805002), the registration memory block contains the following fields:

latestAppVersion
The latest available version of the application installed on the device [16-bit unsigned]

installedAppVersion
Version of the application currently installed on the device [16-bit unsigned]

appId
Application ID of the application installed on the device [max. 50 characters]

5.5.2 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 Cloud 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
Application ID of the application template from which the site should be created [max. 50 characters].

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 Cloudserver 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 Cloud 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.

5.6 File transfer

It is possible to register as many files as you like 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 237). The callback function must be able to handle all file transfer commands (see "File transfer commands" in chapter "Constants" on page 236). The "t.b.d." script supplied with the rapidM2M Toolset contains an example for handling file transfer commands. 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.

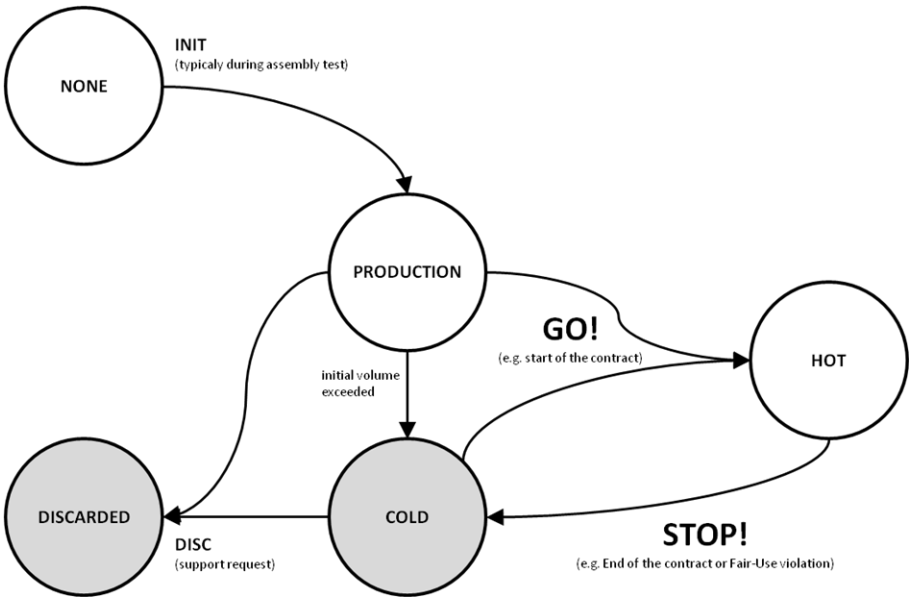
Upon receipt of a file transfer command, a session is started that is automatically terminated after 15sec. if the script is not handling the command correctly. Sessions can only be active one after another to prevent any conflicts.

5.7 Meaning of the SIM state

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

SIM state	Transmission via script	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 script 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 307). 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).

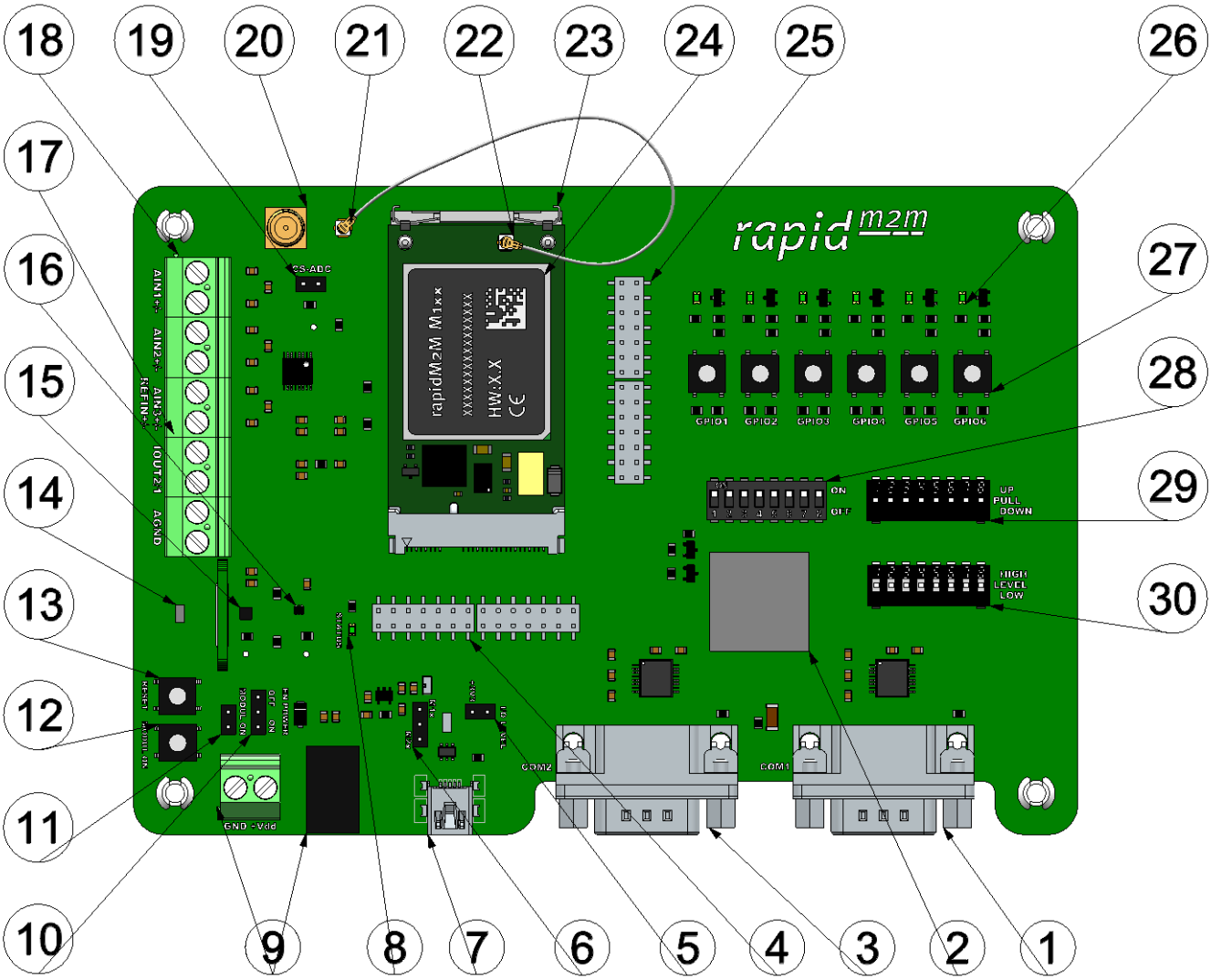


State diagram of the SIM states

6872
6872
6872

Chapter 6 rapidM2M board for developers

6.1 Overview



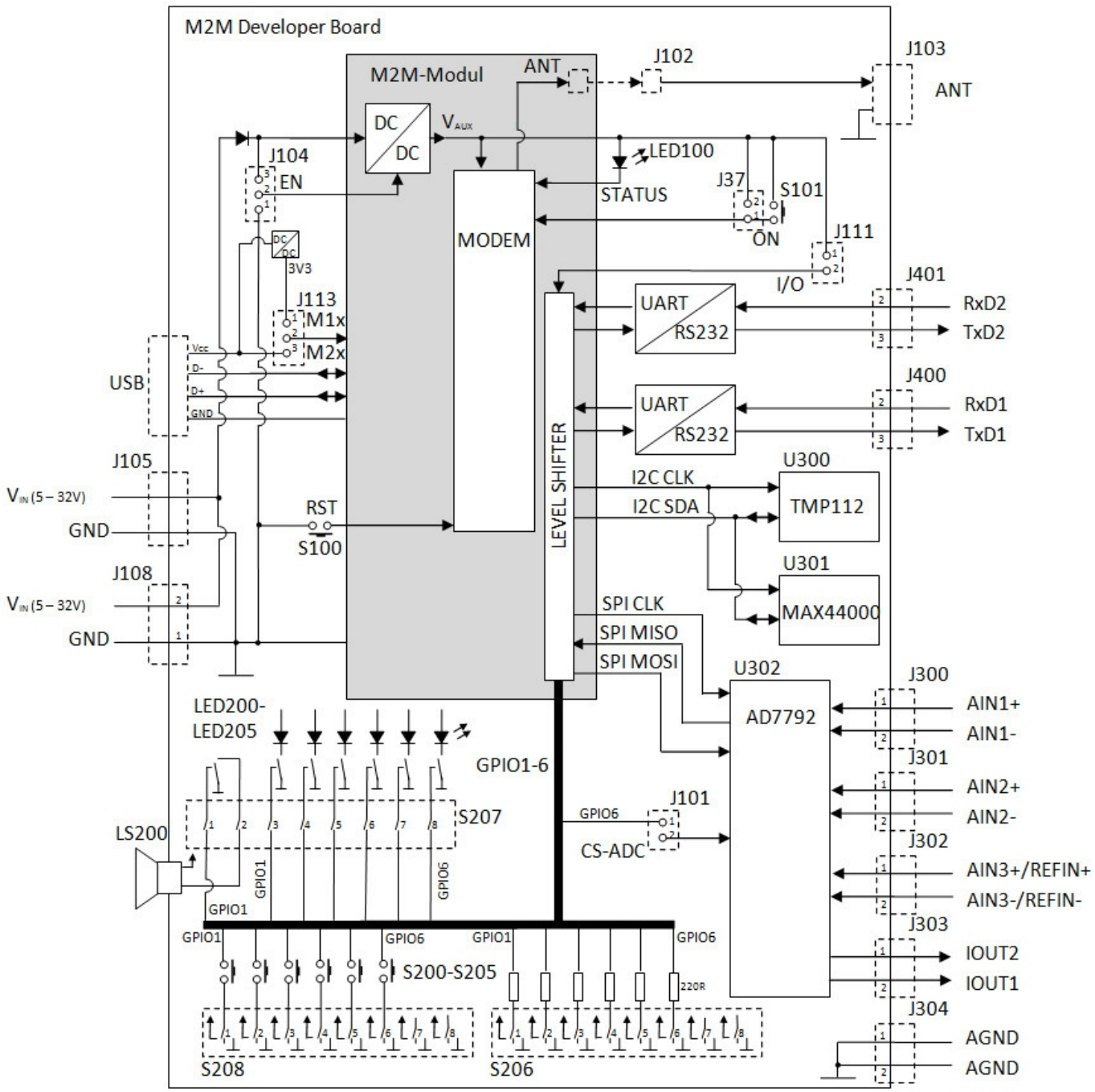
Overview rapidM2M developer package M120

1 COM1 (J400) (see "RS232 interfaces" on page 71)	**16** Temperature sensor (U300) (see "Temperature sensor TMP112 (U300)" on page 83)
2 Buzzer (LS200) (see "Output of digital signals" on page 72)	**17** Constant current outputs (J303 and J304) (see "ADC AD7792 (U302)" on page 82)
3 COM2 (J401) (see "RS232 interfaces" on page 71)	**18** Analogue differential inputs AIN1 to AIN3 (J300-J302) (see "ADC AD7792 (U302)" on page 82)
4 Plug connector to which the odd-numbered pins of the module are connected (J106 and J109) (see "Plug connector to which the odd-numbered pins of the M120 module are connected (J106 and J109)" on page 75)	**19** Jumper for connecting the GPIO6 with the CS line of the ADC (J101) (see "ADC AD7792 (U302)" on page 82)
5 Jumper for supplying the level shifters (J111) (see "Jumper for supplying the level shifters (J111)" on page 75)	**20** SMA antenna connector (J103) (see "Connecting the GSM antenna" on page 95)
6 Jumper for selecting the USB supply for the module (J113) (see "Jumper for selecting the USB supply for the module (J113)" on page 75)	**21** U.FL antenna connector to connect the module with the developer board (J102) (see "Inserting the M120 module" on page 92)
7 Mini-B USB (J402) (see "USB interface" on page 79)	**22** U.FL antenna connector of the module (see "Inserting the M120 module" on page 92)
8 Status LED (LED100) (see "Status LED (LED100)" on page 79)	**23** Holding fixture for the module (see "Holding fixture for the M120 module " on page 84)
9 Supply (DC coaxial connector (J105) or terminal strip (J108)) (see "Supply (J105 or J108) " on page 79)	**24** rapidM2M module M120 (see "rapidM2M module M120 " on page 97)
10 Jumper to activate the energy supply of the module (J104) (see "Jumper to activate the energy supply of the M120 module (J104) " on page 80)	**25** Plug connector to which the even-numbered pins of the module are connected (J107 and J110) (see "Plug connector to which the even-numbered pins of the M120 module are connected (J107 and J110)" on page 77)
11 Jumper to switch on the wireless GSM controller on the module (J37) (see "Jumper to switch on the wireless GSM controller on the rapidM2M module M120 (J37)" on page 80)	**26** LED to display the state of the relevant GPIO (LED200-LED205) (see "Output of digital signals" on page 72)

12 Button to switch on the wireless GSM controller on the module (S101) (see "Button to activate the wireless GSM controller on the rapidM2M module M120 (S101)" on page 80)	**27** Button to simulate a pulse for the relevant GPIO (S200-S205) (see "Input of digital signals" on page 73)
13 Button to reset the wireless GSM controller on the module (S100) (see "Button to reset the wireless GSM controller on the M120 module (S100)" on page 81)	**28** Dip switch (S207) to activate the LEDs for the relevant GPIO and the buzzer (LS200) that is actuated via GPIO1 (see "Output of digital signals" on page 72)
14 IR LED for the proximity sensor (LED300) (see "Ambient light and IR proximity sensor MAX44000 (U301)" on page 84)	**29** Dip switch (S206) to activate the pull-up or pull-down resistors for the relevant GPIO (see "Input of digital signals" on page 73)
15 Ambient light and IR proximity sensor (U301) (see "Ambient light and IR proximity sensor MAX44000 (U301)" on page 84)	**30** Dip switch (S208) to determine whether the relevant GPIO is set to "low" or "high" by pressing the assigned button (see "Input of digital signals" on page 73)

6.1.1 Block diagram

Note: *A detailed block diagram of the M120 module inserted in the rapidM2M board for developers is provided in chapter "Block diagram" on page 100. All of the signals (e.g. SPI, I²C, GPIO, etc.) of the 52pin connector of the M120 module can be accessed via plug connectors J106, J109 (see "odd pin numbers") and J107, J110 (see "even pin numbers") of the rapidM2M board for developers . For the sake of clarity, these plug connectors have not been included in the block diagram.*



Block diagram of the developer board

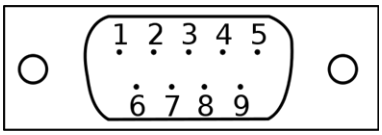
6.1.2 RS232 interfaces

The UART1 of the M120 module can be accessed via the COM1 connector (J400) of the developer board. The UART2 of the M120 module can be accessed via the COM2 connector (J401) of the developer board. Conversion to the RS232 level is completed by means of a level converter located on the rapidM2M board for developers (see "Block diagram" on page 70).

> ***Important note:*** *The RS232 level converters cannot be deactivated via jumpers. The RS232 level converters must be removed if you want to supply the signals directly via the plug connector pins to which the uneven numbers of the M120 module are connected (see "Plug connector to which the odd-numbered pins of the M120 module are connected (J106 and J109)" on page 75).*

Baud rate	300-115200
Stop bits	1, 2
Parity	N, E, O
Data bits	7, 8
Modes	ASCII

The direction of the signals corresponds to that of a DTE (e.g. PC).



9-pin Sub-D(m)

Assignment of the Sub-D connector

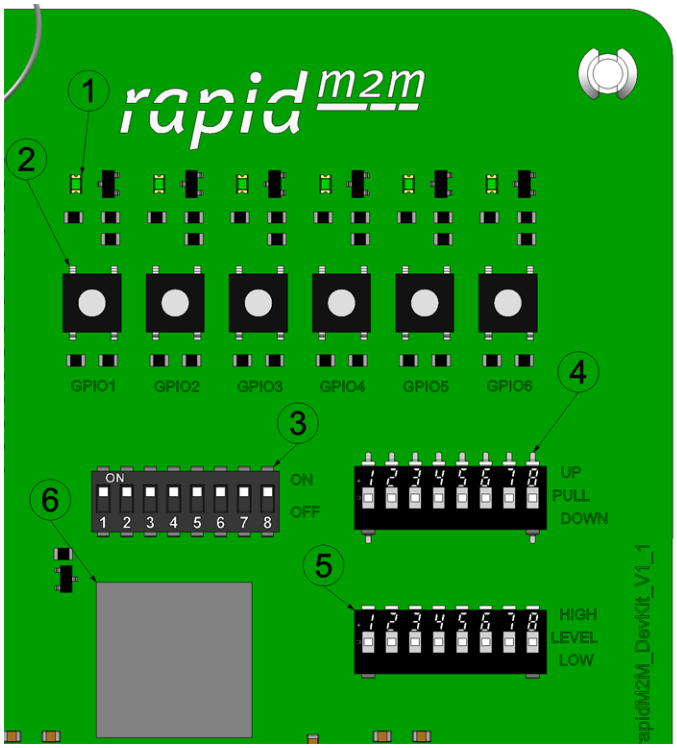
Pin	Signal
1	NC
2	RXD
3	TXD
4	NC
5	GND
6	NC
7	RTS
8	CTS
9	NC

6.1.3 Simulation circuits

Note: A detailed circuit diagram of the simulation circuits is provided in chapter "GPIO" on page 89.

The rapidM2M board for developers comprises 6 identical circuits that can either be used to display the state of the relevant GPIOs via LED (LED200-LED205) or to enter a state via buttons (S200-S205).

The GPIO1 can also be used to activate buzzer LS200.



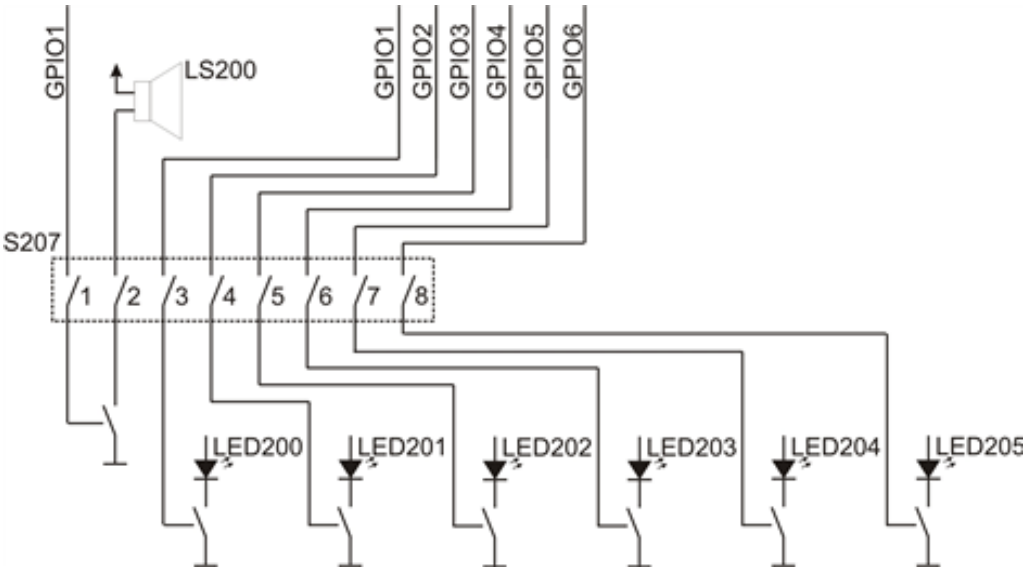
Detailed overview: Simulation circuits

1	LED (LED200) to display the status of GPIO1	4	Dip switch (S206) to activate the pull-up or pull-down resistor for the relevant GPIO
2	Switch (S200) to simulate a pulse for GPIO1	5	Dip switch (S208) to determine whether the relevant GPIO is set to "low" or "high" by pressing the assigned button
3	Dip switch (S207) to activate the LEDs for the relevant GPIO and the buzzer (LS200) that is actuated via GPIO1.	6	Buzzer (LS200)

6.1.3.1 Output of digital signals

To use one of the GPIOs as an output, it must first be configured as an output via the "rM2M_GpioDir()" PAWN script function. The "rM2M_GpioSet()" function can then be used to set the relevant GPIO to "high" or "low". When dip switch S207 is set to "ON", the LED lights up when the associated GPIO is set to "high" or the buzzer emits a sound when the GPIO1 is set to "low".

Note: *The corresponding pull-up or pull-down resistor must be deactivated (centred position of dip switch S206) if you use a GPIO as an output. The output stage of the GPIO is not able to switch the level against the resistor. This means that if pull-up is activated, the output stage cannot set the level to "low".*



Schematic diagram in the event that the GPIOs should be used as outputs

Dip switch S207 to activate the LEDs and the buzzer

1	Connect GPIO1 with the FET to activate the buzzer (LS200)
2	Connect the control FET with the buzzer (LS200)
3	Connect GPIO1 with the FET to activate LED200
4	Connect GPIO2 with the FET to activate LED201
5	Connect GPIO3 with the FET to activate LED202
6	Connect GPIO4 with the FET to activate LED203
7	Connect GPIO5 with the FET to activate LED204
8	Connect GPIO6 with the FET to activate LED205

Note: *Please ensure that GPIO6 is also connected to the CS line of the ADC (AD7792) via jumper "CS-ADC" (J101) (see "Block diagram" on page 70). Remove this jumper if you want to use the simulation circuit for GPIO6 (see "Overview" on page 67).*

6.1.3.2 Input of digital signals

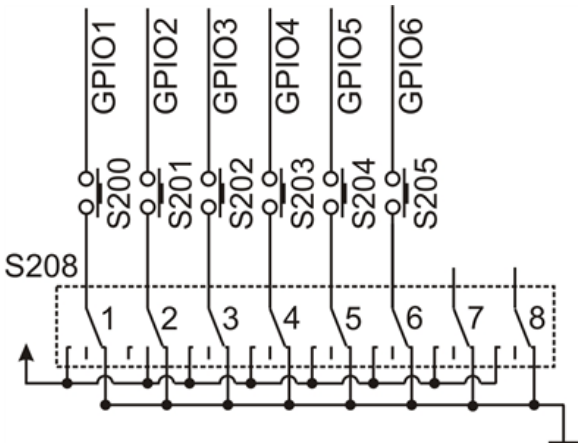
To use one of the GPIOs as an input, it must first be configured as an input via the "rM2M_ GpioDir()" PAWN script function. The level at the GPIO can then be read in with the "rM2M_ GpioGet()" function. Dip switch S208 can be used to determine whether the button (S200-S205) assigned to the GPIO should set the input to "high" or "low" when pressed. The following two tables explain the function of dip switch S208.

Explanation of the switch position of dip switch S208

+	Pressing the button sets the GPIO to "high".
o	---
-	Pressing the button sets the GPIO to "low".

Dip switch S208 to determine whether the GPIO is set to "low" or "high" by pressing the button

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



Schematic diagram: Dip switch S208 to determine whether the GPIO is set to "low" or "high" by pressing the button

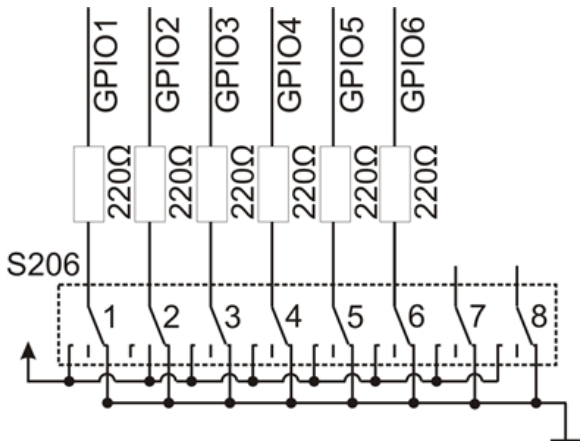
Pull-up and pull-down resistors, as described in the two following tables, can be connected via dip switch S206.

Explanation of the switch position of dip switch S206

+	220Ω pull-up
o	---
-	220Ω pull-down

Dip switch S206 for activating the pull-up or pull-down resistors

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



Schematic diagram: Dip switch S206 to activate the pull-up or pull-down resistors

6.1.4 Jumper for supplying the level shifters (J111)

The module is equipped with level shifters for all of the I/O signals so that the signal level of the M120 module can be adjusted to the surrounding hardware. To complete an adjustment, an additional supply voltage of 2,8...5,5V must be provided for the level shifters by the surrounding hardware. V_{AUX} from the M120 module is supplied back into the module via jumper J111 on the developer board to supply the level shifters (see "Block diagram" on page 70).

> ***Note:*** *Jumper J111 should always be closed.*

6.1.5 Jumper for selecting the USB supply for the module (J113)

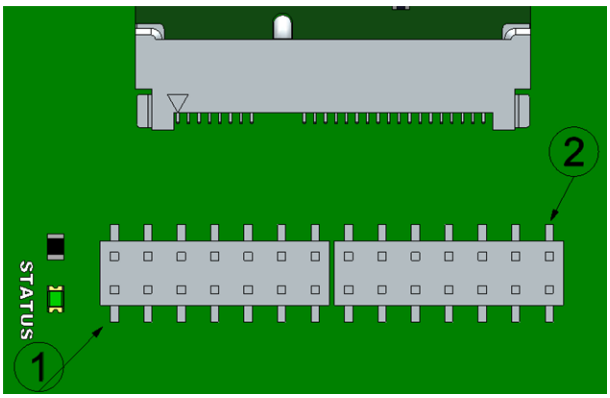
Jumper J113 can be used to determine the USB supply for the module installed in the rapidM2M board for developers (see "Block diagram" on page 70). It is therefore possible to use series "M1x" and "M2x" modules.

Explanation of the selection options for jumper J113

Position	USB supply	Modules
M1x	3,3V	rapidM2M module M11 (300068) rapidM2M module M120 (300087)
M2x	5V	

> ***Note:*** *Before establishing the USB connection between the rapidM2M board for developers and your PC, ensure that jumper J113 is set appropriately for your type of module.*

6.1.6 Plug connector to which the odd-numbered pins of the M120 module are connected (J106 and J109)



Plug connector to which the odd-numbered pins of the M120 module are connected

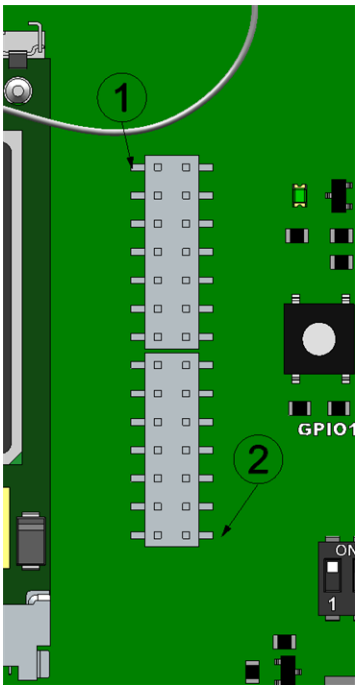
1 Pin 1	**2** Pin 28

Pin	Type	Title	Description
1	I	Module_ON	Pin to switch on the wireless GSM controller on the rapidM2M module M120
2	I	EN_power	Pin to activate the DC/DC converter that generates the internal supply voltage V_{AUX} from the voltage connected to V_{IN}
3	O	LED_status	Open drain output to connect a status LED
4	I	Reset	Pin to reset the wireless GSM controller on the rapidM2M module M120
5-6, 13		GND	Ground
7-8			Reserved for extensions[1)]
9	I	V_{IN}	Supply voltage (4,5...32VDC) for the rapidM2M module M120
10, 15	I/O	V_{AUX}	3,65V voltage output if the rapidM2M module M120 is supplied via V_{IN} Supply voltage input (3,2...4,5V) if no voltage is applied to V_{IN}
11	I/O	GPIO2	GPIO for exchanging signals between M120 and the surrounding hardware
12	I/O	GPIO3	
14	I/O	GPIO1	
16	I	I/O level	2,8...5,5V supply of the level shifters to adjust the signal level of the M120 module to the surrounding hardware
17	I	INT0	Inputs via which the surrounding hardware can trigger interrupts on the rapidM2M module M120
18	I	INT1	
19-20		GND	Ground
21	O	RTS2	RTS line of UART2[2)]
22	I	CTS2	CTS line of UART2[2)]
23	O	TxD2	TxD line of UART2[2)]
24	I	RxD2	RxD line of UART2[2)]
25	O	RTS1	RTS line of UART1[2)]
26	I	CTS1	CTS line of UART1[2)]
27	O	TxD1	TxD line of UART1[2)]
28	I	RxD1	RxD line of UART1[2)]

[1)] No signals must be connected to these pins.

[2)] The RS232 level converters on the developer board must be removed if the signals should be supplied directly via these pins.

6.1.7 Plug connector to which the even-numbered pins of the M120 module are connected (J107 and J110)



Plug connector to which the even-numbered pins of the M120 module are connected

1 Pin 1	**2** Pin 28

Pin	Type	Title	Description
1, 27-28	I	V_{IN}	Supply voltage (4,5...32VDC) for the rapidM2M module M120
2, 18	I/O	V_{AUX}	3,65V voltage output if the rapidM2M module M120 is supplied via V_{IN} Supply voltage input (3,2...4,5V) if no voltage is applied to V_{IN}
3-8			Reserved for extensions[1)]
9-10		GND	Ground
11	I/O	GPIO5	GPIO for exchanging signals between M120 and the surrounding hardware
12	I/O	GPIO4	
13	I	Reset	Pin to reset the wireless GSM controller on the rapidM2M module M120
14		NC	
15	O	I2C_SCL	Clock line of the I^2C interface
16	I/O	I2C_SDA	Data line of the I^2C interface
17	I	USB_3V3	USB supply ***Important note:*** *Only 3,3V are permitted on this pin instead of the usual 5V.*
19	I/O	USB_D-	"Data-" line of the USB interface
20	I/O	USB_D+	"Data+" line of the USB interface
21-22		GND	Ground
23	O	SPI_MOSI	"Master out, slave in" line of the SPI interface
24	I	SPI_MISO	"Master in, slave out" line of the SPI interface
25	O	SPI_CLK	Clock line of the SPI interface
26	I/O	GPIO6	GPIO for exchanging signals between M120 and the surrounding hardware

[1)] No signals must be connected to these pins.

6.1.8 USB interface

The connection to a PC is established via the USB slave interface. It is only designated for the communication with the rapidM2M Toolset . It is not possible to access the USB interface via the PAWN script. The toolset and USB interface are only required during the development phase of a new script. This means that it is not absolutely essential that you include the USB interface in your hardware design. A detailed description of the rapidM2M Toolset is provided in chapter "rapidM2M Toolset " on page 137. The rapidM2M Toolset and can be downloaded free of charge from the following website:

www.microtronics.at/Toolset

Important note: Please note that when you integrate the M120 module in your own hardware, that the rapidM2M module M120 operates with a USB supply of 3,3V . The conversion from 5V to 3,3V is completed by a DC/DC converter that is located on the rapidM2M board for developers (see "Block diagram" on page 70). The jumper for selecting the USB supply (J113) must be set to the "M1x" position for this purpose.

6.1.9 Status LED (LED100)

The status LED lights up for approx. 1sec. after the wireless GSM controller of the M120 module is switched on

6.1.10 Supply (J105 or J108)

The supply of the developer board is either provided via the DC coaxial connector (J105) or the terminal strip (J108). The supply voltage input includes reverse voltage protection. As jumper J37 ("MODULE ON") is closed and jumper J104 ("EN POWER") is set to the "ON" position in the delivered condition, the wireless GSM controller of the M120 module starts up once the energy supply of the developer board has been established and as soon as it has detected the "Module_ ON" signal (approx. 1,5sec. after the power supply is generated). In addition to the plug-in power supply included in the scope of delivery, the power supply units that meet the requirements described in the following chapters can also be used.

6.1.10.1 Terminal strip (J108)

U	5...32VDC
I_{min}	t.b.d. mA@5V t.b.d. mA@12V t.b.d. mA@24V

Assignment of the terminal strip (J108)

1	GND	Ground
2	$+V_{dd}$	Supply voltage:5...32VDC

6.1.10.2 DC coaxial connector (J105)

U	5...32VDC
I_{min}	t.b.d. mA@5V t.b.d. mA@12V t.b.d. mA@24V
External/internal Ø	5,5/2,1mm
Polarity	+internal

6.1.11 Jumper to activate the energy supply of the M120 module (J104)

If jumper J104 is open or pin 1 and pin 2 are connected by means of a jumper, the DC/DC converter of the M120 module is inactive and the internal supply of the module is thus switched off.

If pin 2 and pin 3 are connected, the DC/DC converter is activated. Due to the soft start of the converter, it takes 50ms until the internal supply voltage of the M120 module has reached the maximum value.

6.1.12 Button to activate the wireless GSM controller on the rapidM2M module M120 (S101)

50ms (soft start) after the energy supply of the M120 module has been activated by means of jumper J104 (see "Jumper to activate the energy supply of the M120 module (J104) " on page 80), the wireless GSM controller of the M120 module can be switched on by pressing and holding button S101 for 1,5sec. . The reset button S100 must not be activated during the start phase. The status LED (LED100) lights up for approx. 1sec. during the boot process

The current firmware of the M120 module does not provide the means to switch off the wireless GSM controller.

A detailed description of the timings is provided in "ENCPU signal" on page 106.

6.1.13 Jumper to switch on the wireless GSM controller on the rapidM2M module M120 (J37)

Jumper J37 is switched in parallel to button S101 to switch on the wireless GSM controller (see "Block diagram" on page 70). This means that jumper J37 can be set instead of switching on the wireless GSM controller on the rapidM2M module M120 by pressing the buttonrapidM2M module M120 . Provided jumper J104 (see "Jumper to activate the energy supply of the M120 module (J104) " on page 80) is set to the "ON" position, the wireless GSM controller of the M120 module will then start as soon as the energy supply is established and it has detected the "Module_ON" signal (approx. 1,5sec. after the energy supply is established). The reset button S100 must not be activated during the start phase. The status LED (LED100) lights up for approx. 1sec. during the boot process

The current firmware of the M120 module does not provide the means to switch off the wireless GSM controller.

A detailed description of the timings is provided in "ENCPU signal" on page 106.

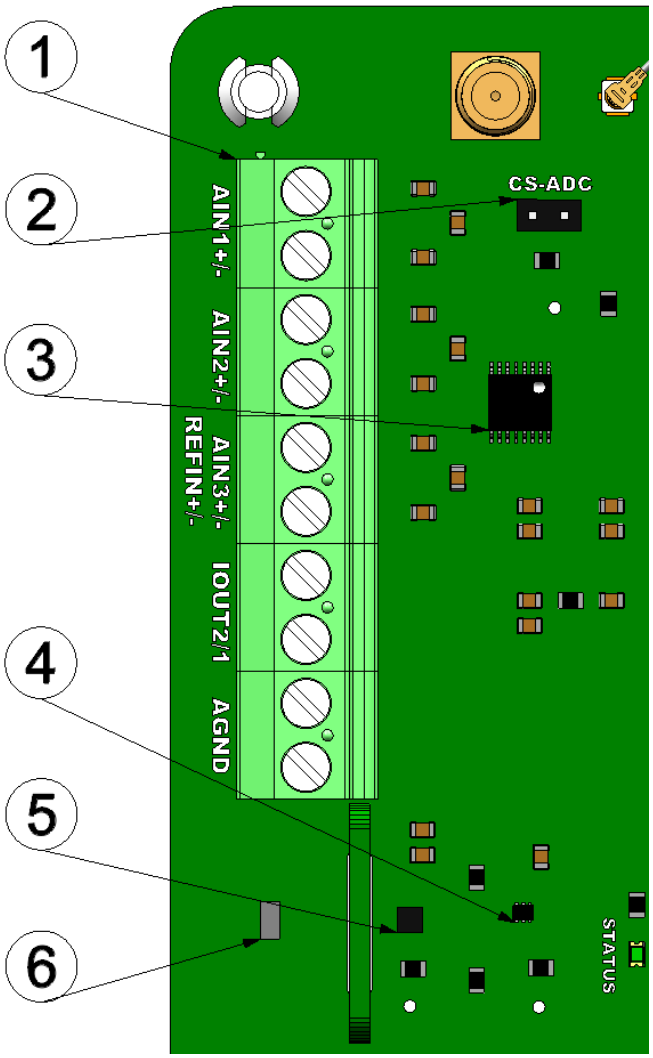
6.1.14 Button to reset the wireless GSM controller on the M120 module (S100)

This button should only be used as an emergency reset, as the wireless GSM controller independently completes a reset process during the boot process. However, if a reset is still required, button S100 must be pressed briefly (200µs). The wireless GSM controller of the M120 module must then be switched back on via button S101 (see "Button to activate the wireless GSM controller on the rapidM2M module M120 (S101)" on page 80).

A detailed description of the timings is provided in "Reset signal" on page 107.

6.1.15 Sensors

There are three sensor chips on the rapidM2M board for developers that, in conjunction with the script examples (see "Example" on page 274), are designed to demonstrate the functionality and integration of the M120 module. The temperature sensor TMP112 (U300) and the ambient light/IR proximity sensor MAX44000 (U301) are connected to the rapidM2M module M120 via the I^2C bus. The ADC AD7792 (U302) is connected to the rapidM2M module M120 via the SPI bus.



Detailed overview: Sensors

1	Terminal strips J300-J304: Differential inputs and constant current outputs of AD7792 (U302)	4	Temperature sensor TMP112 (U300)
2	Jumper for connecting the GPIO6 with the CS line of the ADC (J101)	5	Ambient light and IR proximity sensor MAX44000 (U301)
3	ADC AD7792 (U302)	6	IR LED for the proximity sensor MAX44000 (LED300)

6.1.15.1 ADC AD7792 (U302)

The ADC AD7792 (U302) provides 3 Differential inputs and 2 constant current outputs. There is also an option to use an external reference voltage. Although this does engage one of the differential inputs. The "spi.p" script supplied with the rapidM2M Toolset (see tbd) comprises an example for the use of the AD7792 (U302).

> ***Note:*** *Jumper "CS-ADC" (J101) that connects the CS line of the AD7792 (U302) with GPIO6 must be closed when using AD7792 (U302).*

Differential inputs

Resolution	16 Bit
Input voltage range	V_{ref} / gain
Internal reference voltage	1,17V
External reference voltage	nominal 2,5V, max3,3V
Gain	1-128

Constant current outputs

Output current	10/210/1000µA

Terminal strips J300-J304 (Analogue differential inputs and Constant current outputs)

J300	1	AIN1+	Differential input 1
	2	AIN1-	
J301	1	AIN2+	Differential input 2
	2	AIN2-	
J302	1	AIN3+ / REFIN+	Differential input 3 / input for the external reference voltage
	2	AIN3- / REFIN-	
J303	1	IOUT2	Constant current output 2
	2	IOUT1	Constant current output 1
J304	1	AGND	Analog ground
	2	AGND	Analog ground

Further technical details and functional descriptions are included in the data sheet provided by the manufacturer Analog Devices. The manufacturer's designation of the ADC is "AD7792BRUZ". In the event of any inconsistencies between the technical data provided in this manual and that in the data sheet provided by the chip manufacturer, Analog Devices, the later shall always be deemed to be valid.

6.1.15.2 Temperature sensor TMP112 (U300)

The "i2c.p" script provided with the rapidM2M Toolset (see tbd) includes an example for the use of both sensors (TMP112 and MAX44000) connected to the I^2C bus.

Measurement range	-40...+125°C
Resolution	0,0625°C

Further technical details and functional descriptions are included in the data sheet provided by the manufacturer Texas Instruments. The manufacturer's designation of the temperature sensor is "TMP112AIDRLT". In the event of any inconsistencies between the technical data provided in this manual and that in the data sheet provided by the chip manufacturer, Texas Instruments, the later shall always be deemed to be valid.

6.1.15.3 Ambient light and IR proximity sensor MAX44000 (U301)

The MAX44000 (U301) is a combined ambient light and IR proximity sensor with integrated driver stage for the connection of an IR LED. The "i2c.p" script provided with the rapidM2M Toolset (see tbd) includes an example for the use of both sensors (TMP112 and MAX44000) connected to the I^2C bus.

Measurement range	0,03...65535lux

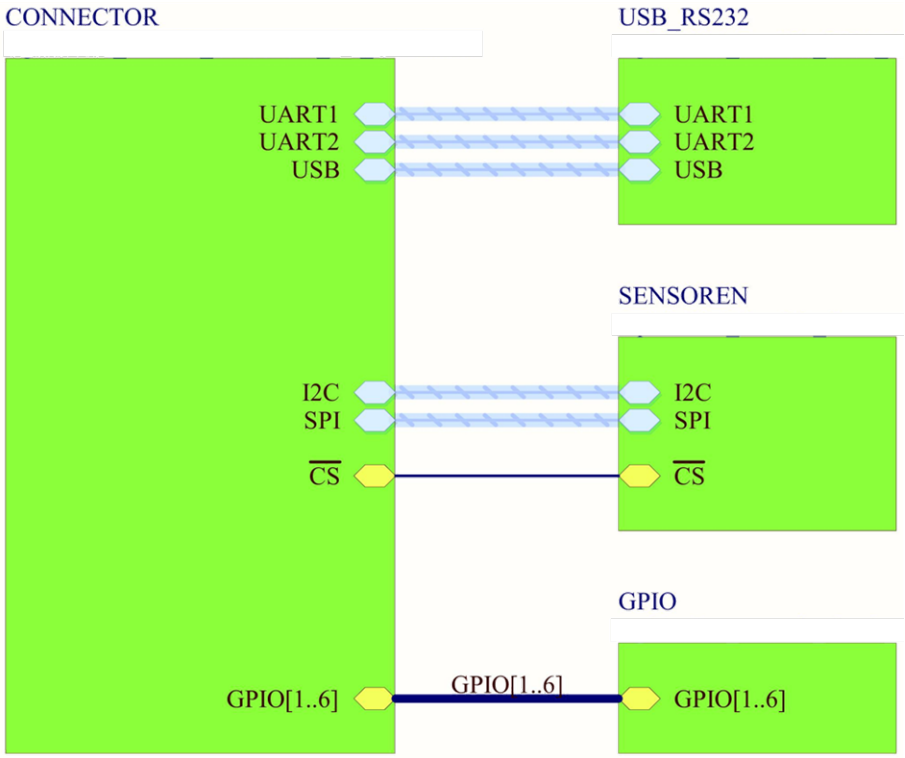
Further technical details and functional descriptions are included in the data sheet provided by the manufacturer Maxim. The manufacturer's designation for the ambient light and IR proximity sensor is "MAX44000GDT+". In the event of any inconsistencies between the technical data provided in this manual and that in the data sheet provided by the chip manufacturer, Maxim, the later shall always be deemed to be valid.

6.1.16 Holding fixture for the M120 module

The holding fixture for the M120 module used on the rapidM2M board for developers is manufactured by JAE Connectors. The manufacturer's designation for this connector is "MM60-52B1-G1-R850".

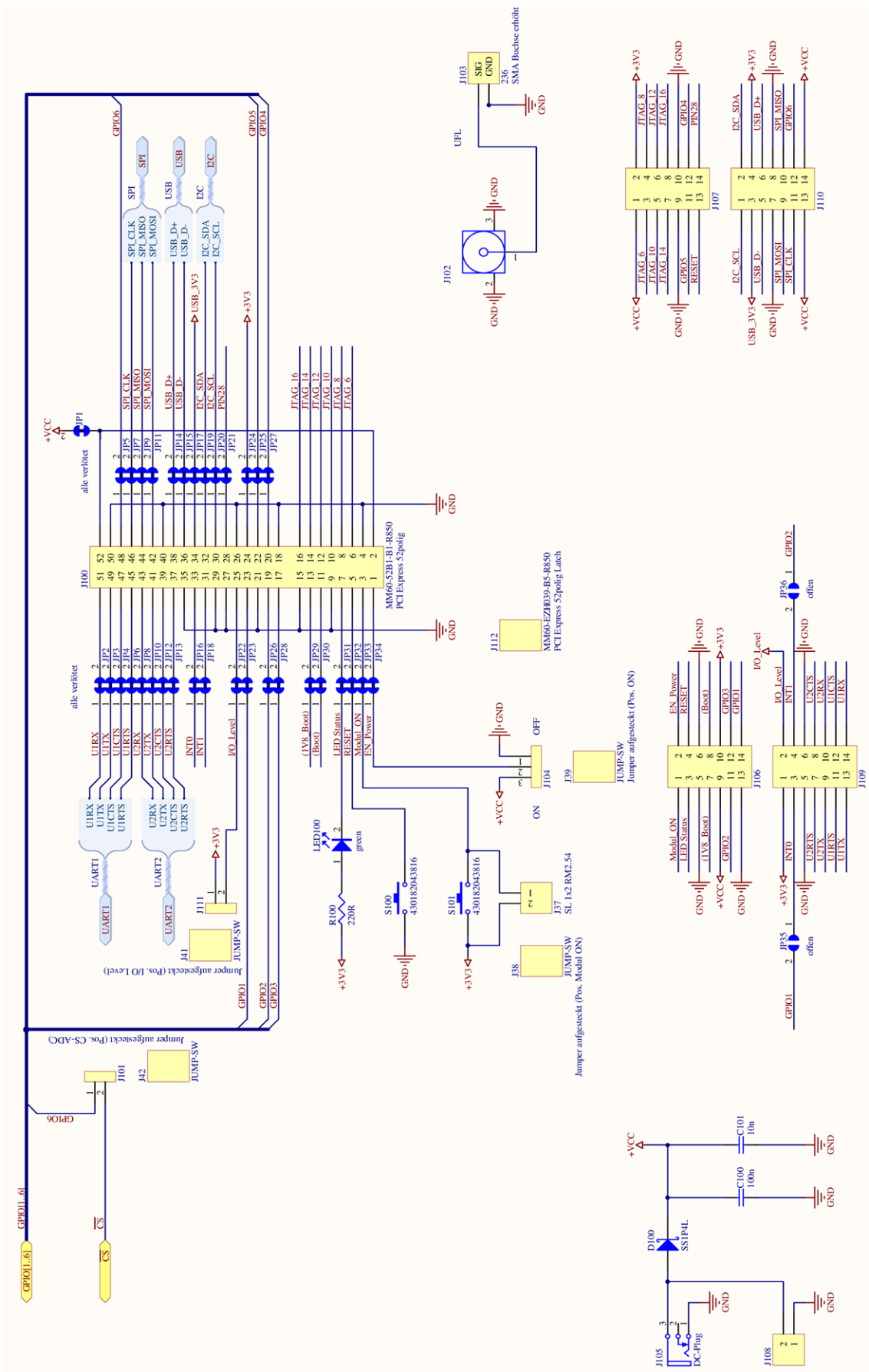
6.2 Circuit diagram

6.2.1 Overview

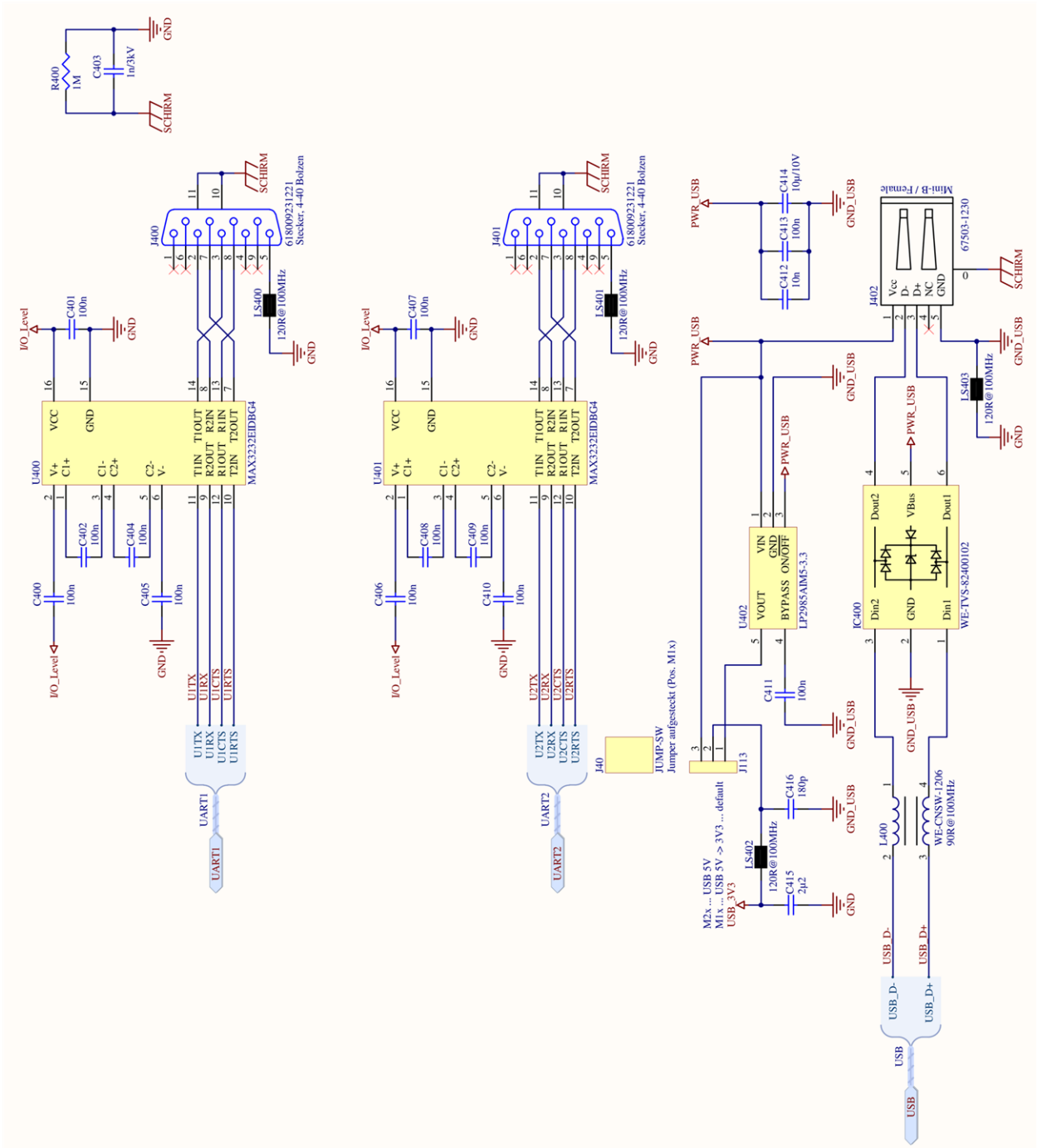


Overview of the circuit diagram of the developer board

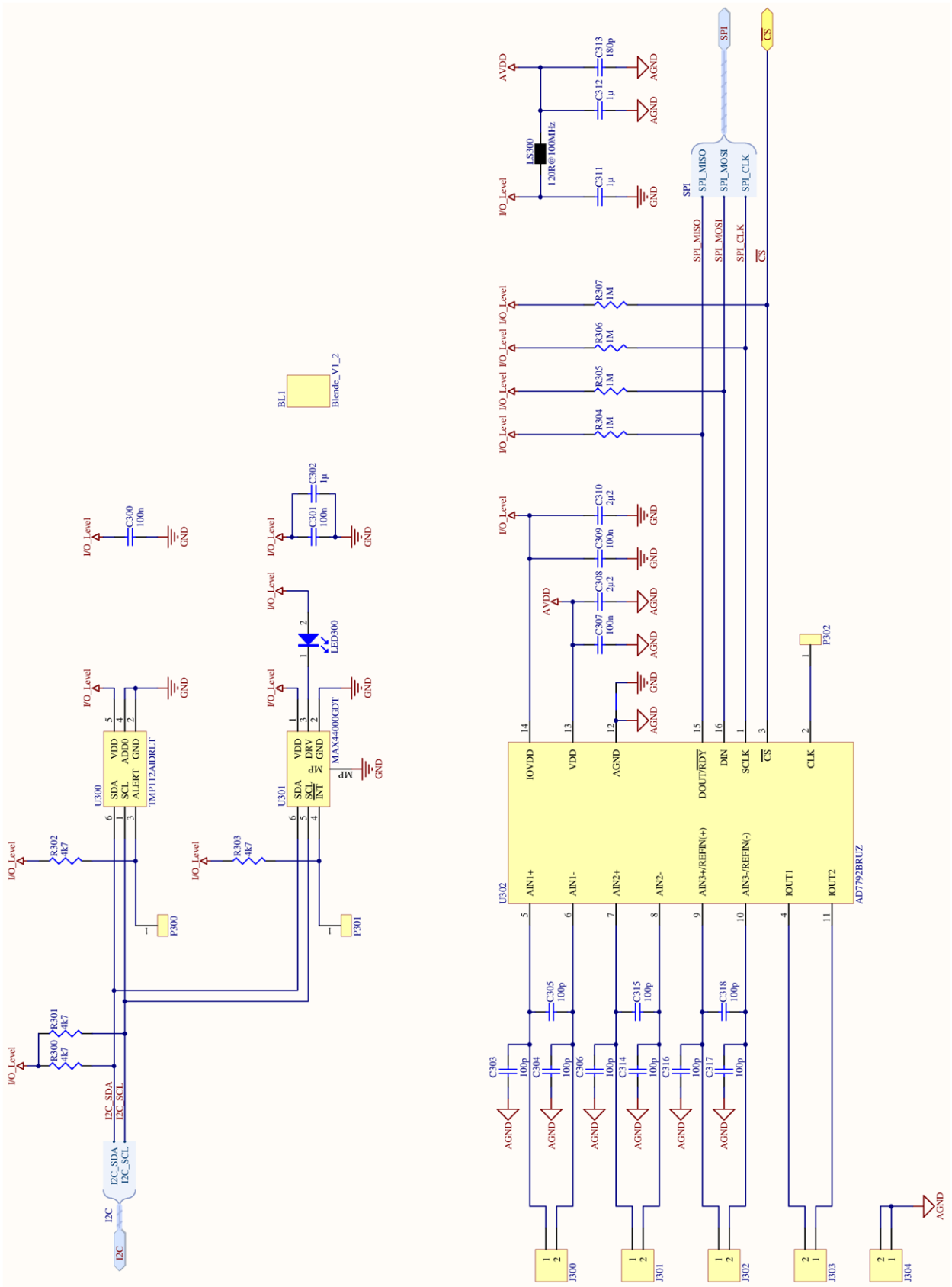
6.2.2 Connector



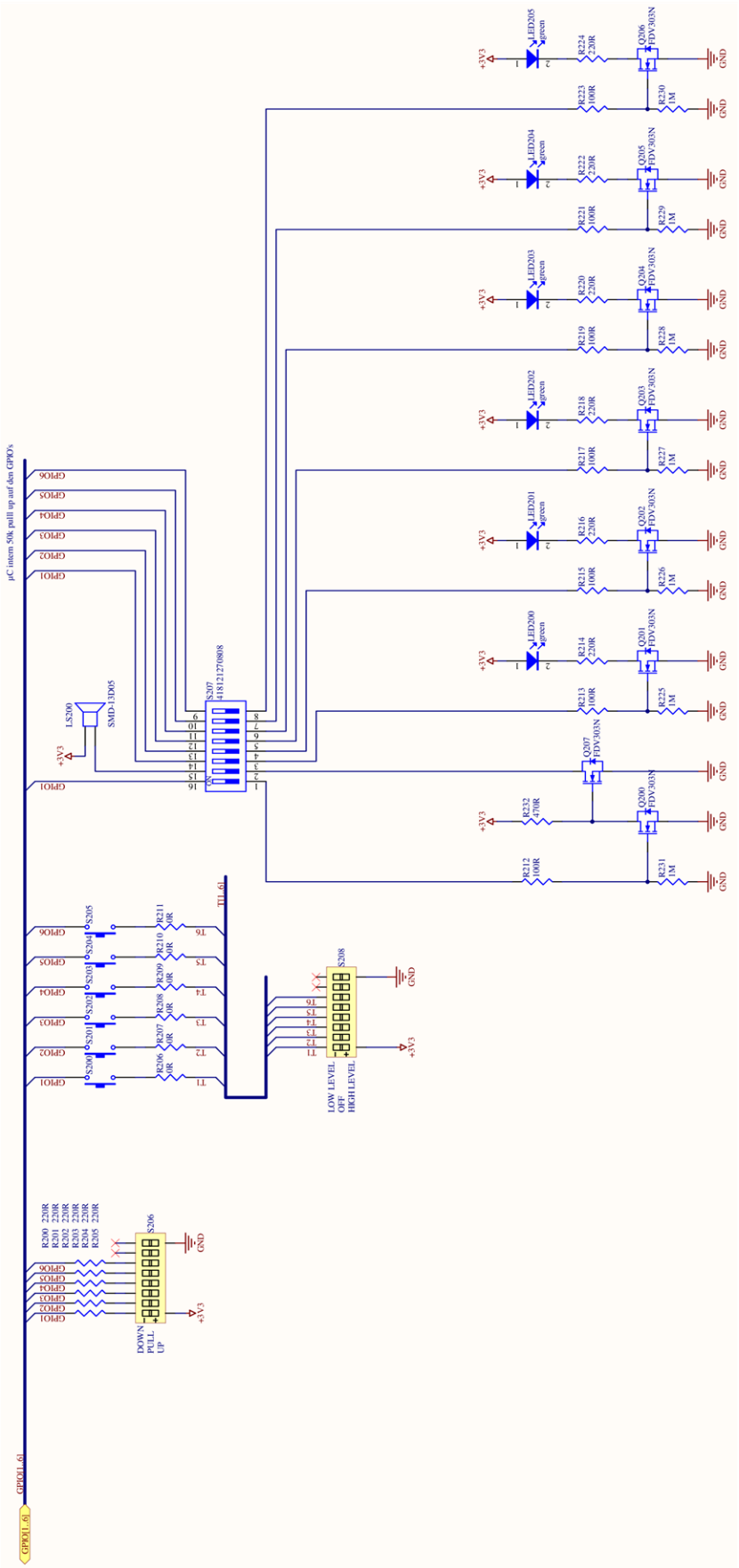
6.2.3 USB RS232



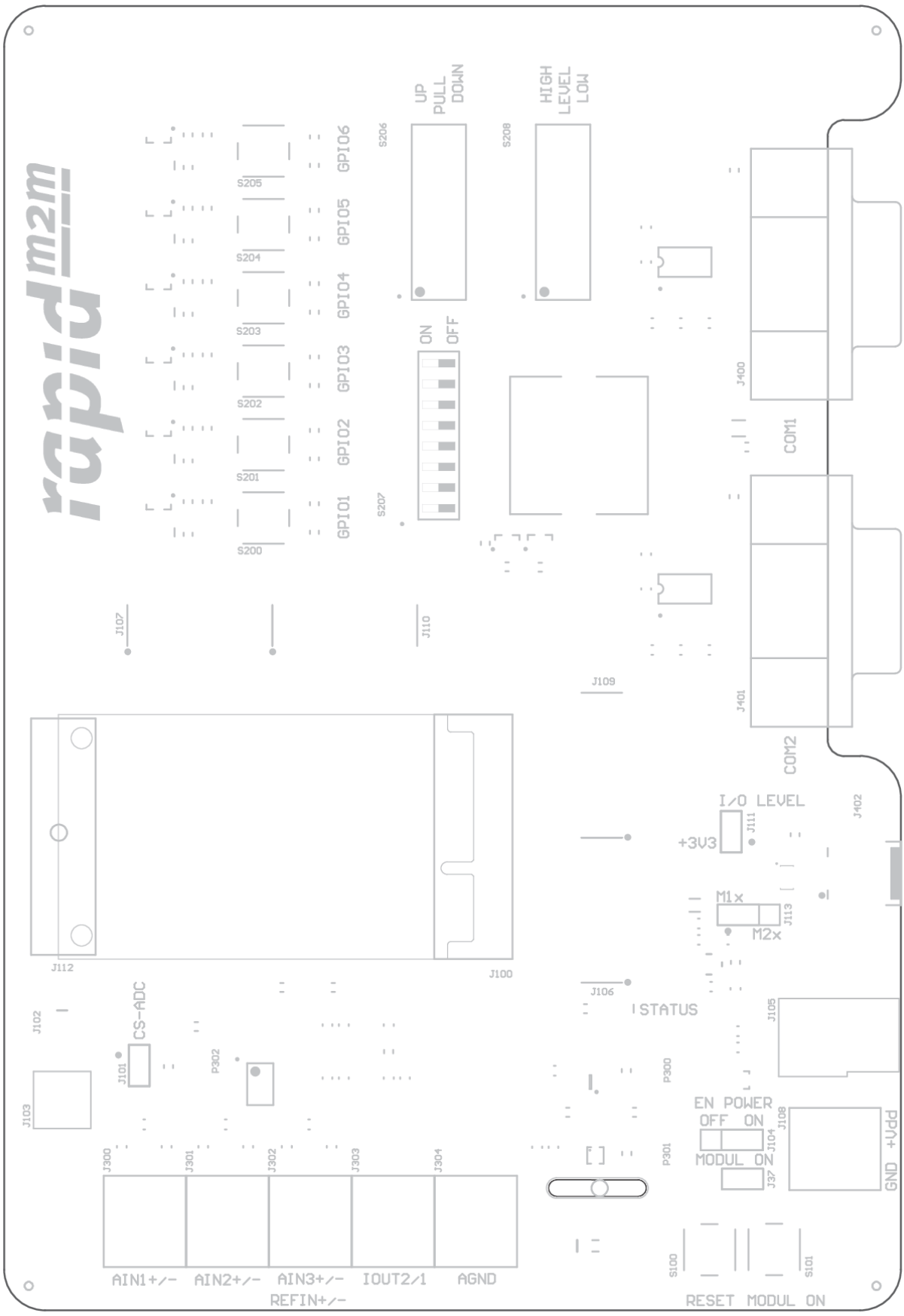
6.2.4 Sensors


U300
TMP112AIDRLT
U301
MAX44006GDT
U302
AD7792BRUZ
LED300
BL1
Blende_V1_2
P300
P301
P302
J300
J301
J302
J303
J304
I2C
SPI
I/O_Level
AVDD
AGND
GND

6.2.5 GPIO


µC intern 50k pull up auf den GPIO's
GPIO6
GPIO5
GPIO4
GPIO3
GPIO2
GPIO1
GPIO1..6
LS200
SMD-3D05
+3V3
S207
41812127080S
TTL 6V
S208
S206
LOW LEVEL
OFF
HIGH LEVEL
DOWN
PULL
UP
GND
LED205
LED204
LED203
LED202
LED201
LED200
green
Q206
Q205
Q204
Q203
Q202
Q201
Q207
Q200
FDV303N
R224
R222
R220
R218
R216
R214
R232
R212
220R
470R
100R
R223
R221
R219
R217
R215
R213
R230
R229
R228
R227
R226
R225
R231
1M
R211
R210
R209
R208
R207
R206
10R
R200
R201
R202
R203
R204
R205

6.3 Layout


rapid m2m
UP
PULL
DOWN
HIGH
LEVEL
LOW
ON
OFF
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
S200
S201
S202
S203
S204
S205
S206
S207
S208
J100
J101
J102
J103
J104
J105
J106
J107
J108
J109
J110
J111
J112
J113
J300
J301
J302
J303
J304
J400
J401
J402
J37
P300
P301
P302
S100
S101
COM1
COM2
I/O LEVEL
+3V3
M1x
M2x
CS-ADC
STATUS
EN POWER
OFF ON
MODUL ON
GND +Vdd
AIN1+/-
AIN2+/-
AIN3+/-
IOUT2/1
AGND
REFIN+/-
RESET MODUL ON

6.4 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.

6.4.1 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.

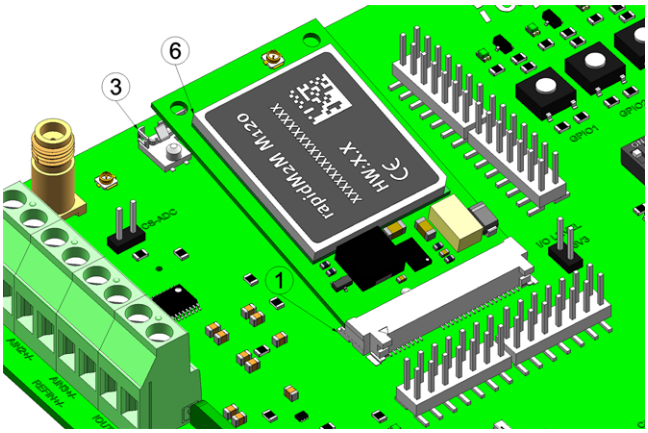
The sensitive electronic components inside the device can be damaged by static electricity, which can impair the device performance or even cause the device to fail. The manufacturer recommends the following steps to prevent any damage to the device caused by electrostatic discharges:

- 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.

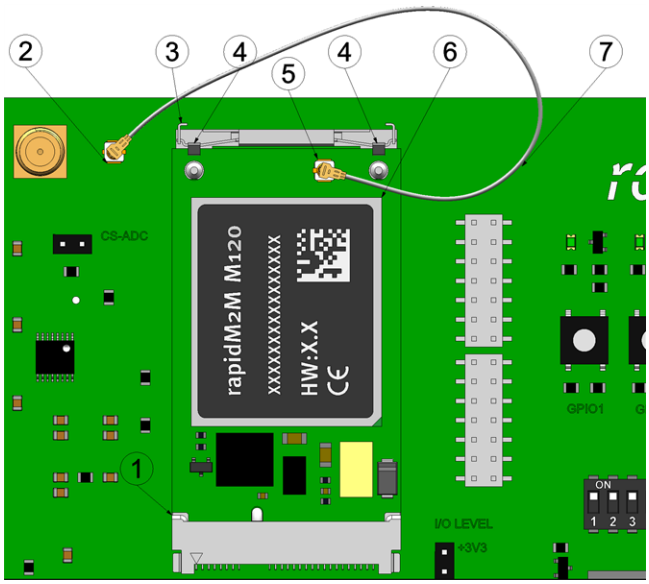
6.4.2 Inserting the M120 module

Important note:

- *The M120 module must be inserted in the de-energised state.*
- *Improper handling can lead to injuries and/or damage to the instruments!*



Inserting the M120 module - step 1



Inserting the M120 module - step 2

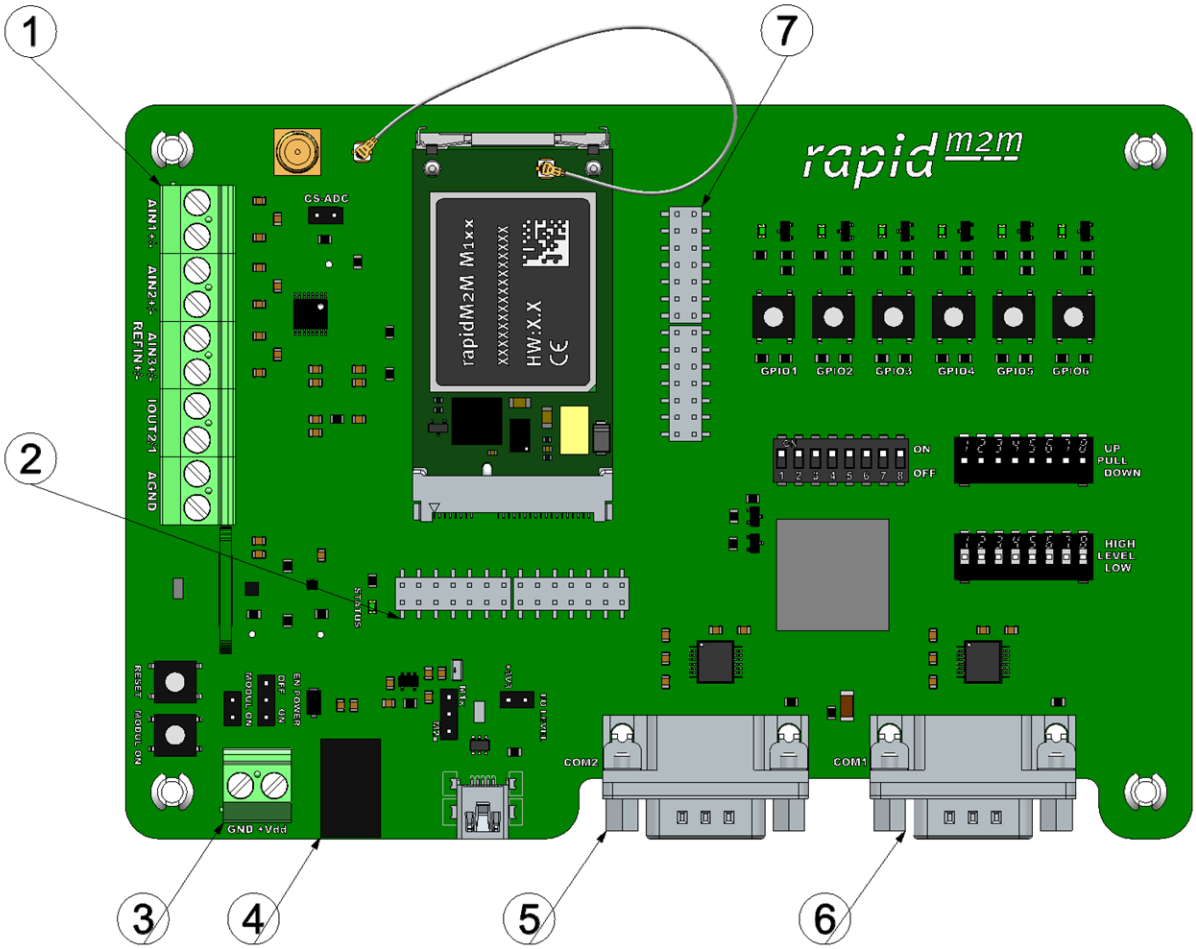
1	52pin plug connector (mechanically compatible with "PCI express mini card")	5	U.FL antenna connector of the M120 module
2	U.FL antenna connector of the developer board (J102)	6	M120 module
3	Locking device	7	U.FL antenna cable
4	Locking device hooks		

1. Insert the M120 module (6) in the 52pin plug connector (1) of the developer board in accordance with the Figure "Inserting the M120 module - step 1 " on page 92. During this process, the module must be inserted in such a way that the type plate is visible once it is locked in place.
2. Apply a slight pressure to the side of the M120 module positioned the furthest away from the 52pin plug connector (1) until the module clicks into the locking device (3). To release the module, push both of the hooks on the locking device (4) backwards.
3. Then connect the U.FL antenna connector of the M120 module (5) with the U.FL antenna connector of the developer board (2) using the U.FL antenna cable included in the scope of supply.

6.4.3 Connecting the sensors, actuators and power supply

Important note:

- *All wiring work must be performed in the de-energised state.*
- *Comply with existing legal and/or operational directives.*
- *Improper handling can cause injuries and/or damage to the instruments!*



Connecting the sensor, actuators and power supply

1	Terminal strips J300-J304: Differential inputs and constant current outputs of AD7792 (U302)	5	COM2: 9-pin D-Sub male (J400)
2	Plug connector (J106 and J109) to which the odd-numbered pins of the M120 module are connected	6	COM1: 9-pin D-Sub male (J401)
3	Power supply: Terminal strip (J108)	7	Plug connector (J107 and J110) to which the even-numbered pins of the M120 module are connected
4	Power supply: DC coaxial connector (J105) 5,5/2,1mm (external/internal diameter)		

Terminal strips J300-J304 (Analogue differential inputs and Constant current outputs)

J300	1	AIN1+	Differential input 1
	2	AIN1-	
J301	1	AIN2+	Differential input 2
	2	AIN2-	
J302	1	AIN3+ / REFIN+	Differential input 3 / input for the external reference voltage
	2	AIN3- / REFIN-	
J303	1	IOUT2	Constant current output 2
	2	IOUT1	Constant current output 1
J304	1	AGND	Analog ground
	2	AGND	Analog ground

Assignment of the terminal strip (J108)

1	GND	Ground
2	$+V_{dd}$	Supply voltage: 5...32VDC

Assignment of the DC coaxial connector (J105) 5,5/2,1mm (external/internal diameter)

External contact	GND	Ground
Internal contact	$+V_{dd}$	Supply voltage: 5...32VDC

1. First of all, connect your sensors and actuators to the rapidM2M board for developers . Ensure that no current is present!

   Note: *Adapters may be required to establish a connection to COM1 (J401) or COM2 (J400). A selection of available adapters is specified in the chapter "Spare parts and accessories" (see "Spare parts and accessories" on page 297).*

2. Connect the antenna (see "Connecting the GSM antenna" on page 95).
3. Ensure that the jumper for supplying the level shifters (J111) is closed and that pin 2 and pin 3 of the jumper used to activate the power supply (J104) are connected ("ON" position) (see "Overview" on page 67).
4. The rapidM2M board for developers can either be powered via the DC coaxial connector (4) or the terminal strip (3). Either use the plug-in power supply included in the scope of delivery or any power supply unit with a voltage of 5...32VDC . If the developer board is to be powered via the terminal strip (3), the cables must be connected to the terminal strip (3) in a de-energised state.

5. Now switch on the power supply of the sensors, actuators and developer board.

   If the jumper for switching on the wireless GSM controller (J37 "MODULE ON") is closed (delivery status, see "Overview" on page 67), the controller of the M120 module starts up once the energy supply of the developer board has been established and as soon as it has detected the "Module_ON" signal (approx. 1,5sec. after the power supply is generated). The status LED should then light up for approx. 1sec. to indicate the boot process (see "Status LED (LED100)" on page 79). The rapidM2M developer package M120 then logs itself into the network and establishes a connection to the Cloud server.

The following step is only necessary, if the jumper for switching on the wireless GSM controller (J37 "MODULE ON") is not closed.

6. Then press the button to switch on the wireless GSM controller S101 (see "Button to activate the wireless GSM controller on the rapidM2M module M120 (S101)" on page 80). The status LED should then light up for approx. 1sec. to indicate the boot process (see "Status LED (LED100)" on page 79). The rapidM2M developer package M120 then logs itself into the network and establishes a connection to the Cloud server.

The following step is not mandatory.

7. Check whether the connection to the Cloud server has worked correctly (see "Testing communication with the device" on page 185).

6.4.4 Connecting the GSM antenna

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

The Magnetic antenna SMA-M 2,5m (300085) included in the scope of supply is connected directly to the antenna connector (see "Overview" on page 67) of the developer board.

If the 2.5 m connection cable of the Magnetic antenna SMA-M 2,5m (300085) is not long enough, you can use a 2.5 m Extension cable for antenna SMA-M/SMA-F 2,5m (206.807).

1. Ensure that the rapidM2M board for developers is de-energised.
2. If you need an antenna extension, connect it to the antenna first.
3. Connect the antenna extension or antenna directly to the antenna connector of the developer board (see "Overview" on page 67).

   > ***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 the antenna manually.*

4. Switch the power supply of the developer board back on.

The following step is only necessary, if the jumper for switching on the wireless GSM controller (J37 "MODULE ON") is not closed.

5. Then press the button to switch on the wireless GSM controller S101 (see "Button to activate the wireless GSM controller on the rapidM2M module M120 (S101)" on page 80).

The following step is not mandatory.

6. Check whether the connection to the Cloud server has worked correctly (see"Testing communication with the device" on page 185).

Chapter 7 rapidM2M module M120

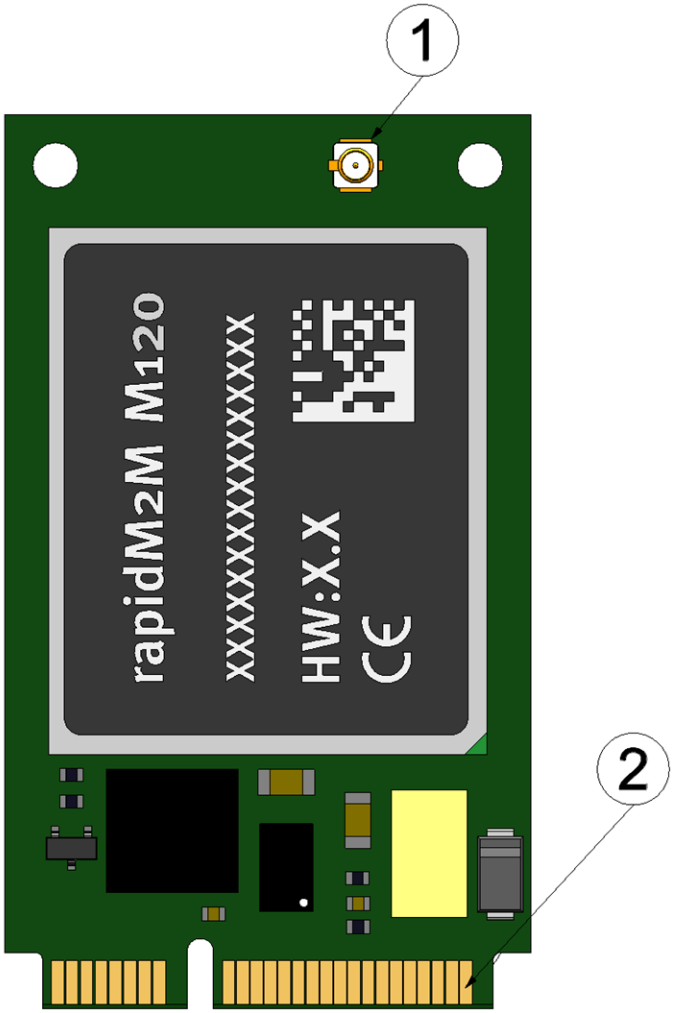
7.1 Technical data

Voltage supply	Supply via V_{IN}: 4,5...32VDC , max. 100mA @ 12V DC[1)] or Supply via V_{AUX}: 3,2...4,5V , max. 294mA @ 3,8V DC[1)] Additional information is provided in "Technical details about the energy supply" on page 105.
Output voltage (only when supplied via V_{IN})	3,65V , max. 300mA
Input range of the level converters for I/Os	2,8...5,5V Additional information is provided in "Technical details about the level shifters" on page 107.
Housing	Weight: 11g Interface: 52pin connector strip Dimensions (WHD): 30 x 5,5 x 51mm (without antenna)
Operating temperature	-20...+60°C, 15...90%rH non-condensing
Storage temperature	-40...+85°C
Display	Open drain output to connect a status LED Additional information is provided in "Status LED (LED100)" on page 79.
Antenna connector	U.FL

[1)]The values were determined at a GSM signal strength of -60dBm. Higher current values are possible during an active GSM data transmission. The peak current is dependent on the GSM signal strength, amount of usage data and the transmission speed.

Serial interfaces	2 x 4-wire UART • Baud rate: 300-115200 • Stop bits: 1, 2 • Parity:N, E, O • Data bits: 7, 8 1 x SPI • Mode: SPI master • Adjustable clock polarity/phase • Clock frequency: 102kHz...13MHz • Interface: 4-wire 1 x I^2C • Mode: Master • Clock frequency: 100kHz...400kHz Additional information is provided in: • "UART" on page 108 • "SPI" on page 109 • "I2C" on page 110
USB interface	1 x mini-B USB 2.0 slave for establishing a connection to a PC. The rapidM2M Toolset must be installed on the PC to enable communication with the rapidM2M module M120 . Additional information is provided in "USB interface" on page 79.
I/O lines	6 x GPIO 2 x interrupt inputs
Data memory	3 x 349.525 Byte internal flash memory The size of the data records is variable (max. 1024 Byte) and is determined by the script created by the user. The system-related overhead is 10 Byteper data record.
Configuration memory	10 independent blocks each with 10kB
Registration memory	4 blocks each with 1kB and pre-defined applications for storing device-specific data Additional information is provided in "Registration memory blocks" on page 62.
Memory for the PAWN binary	256kB (uncompressed size) Additional information is provided in "Memory organisation" on page 103.
Data transmission	By means of GSM/GPRS quad-band modem to the respective Cloud server
SIM	The rapidM2M module M120 is equipped with an integrated SIM chip.

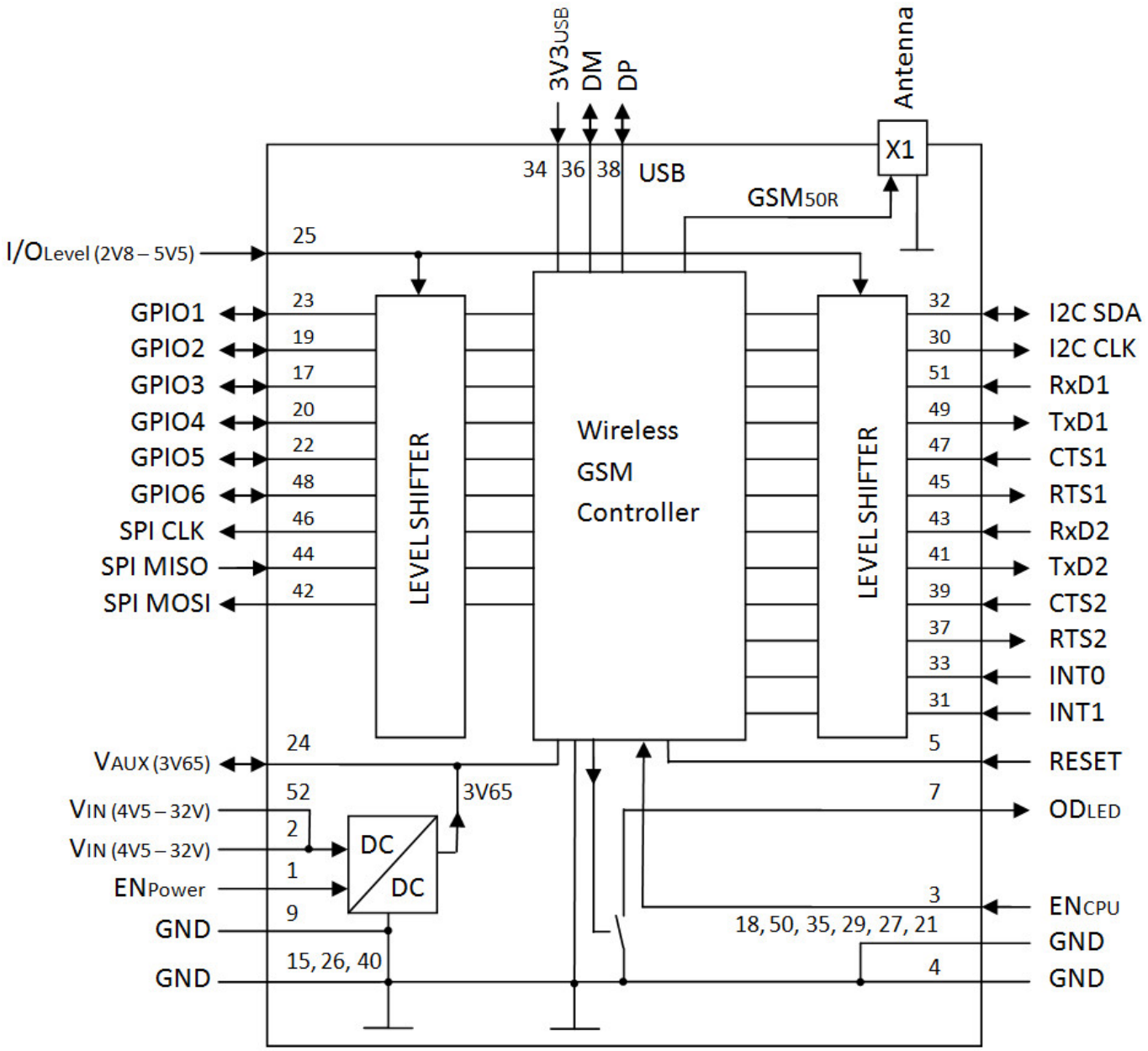
7.2 Overview



Overview M120 module

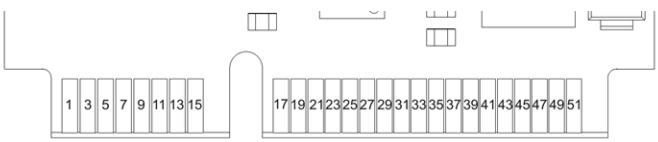
1 U.FL antenna connector (X1)	**2** 52pin connector strip (X2)

7.2.1 Block diagram

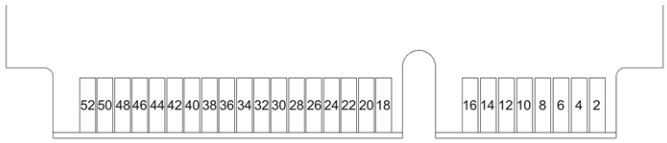


Block diagram of the M120 module

7.2.2 52pin connector strip (X2)



Connector strip (top view)

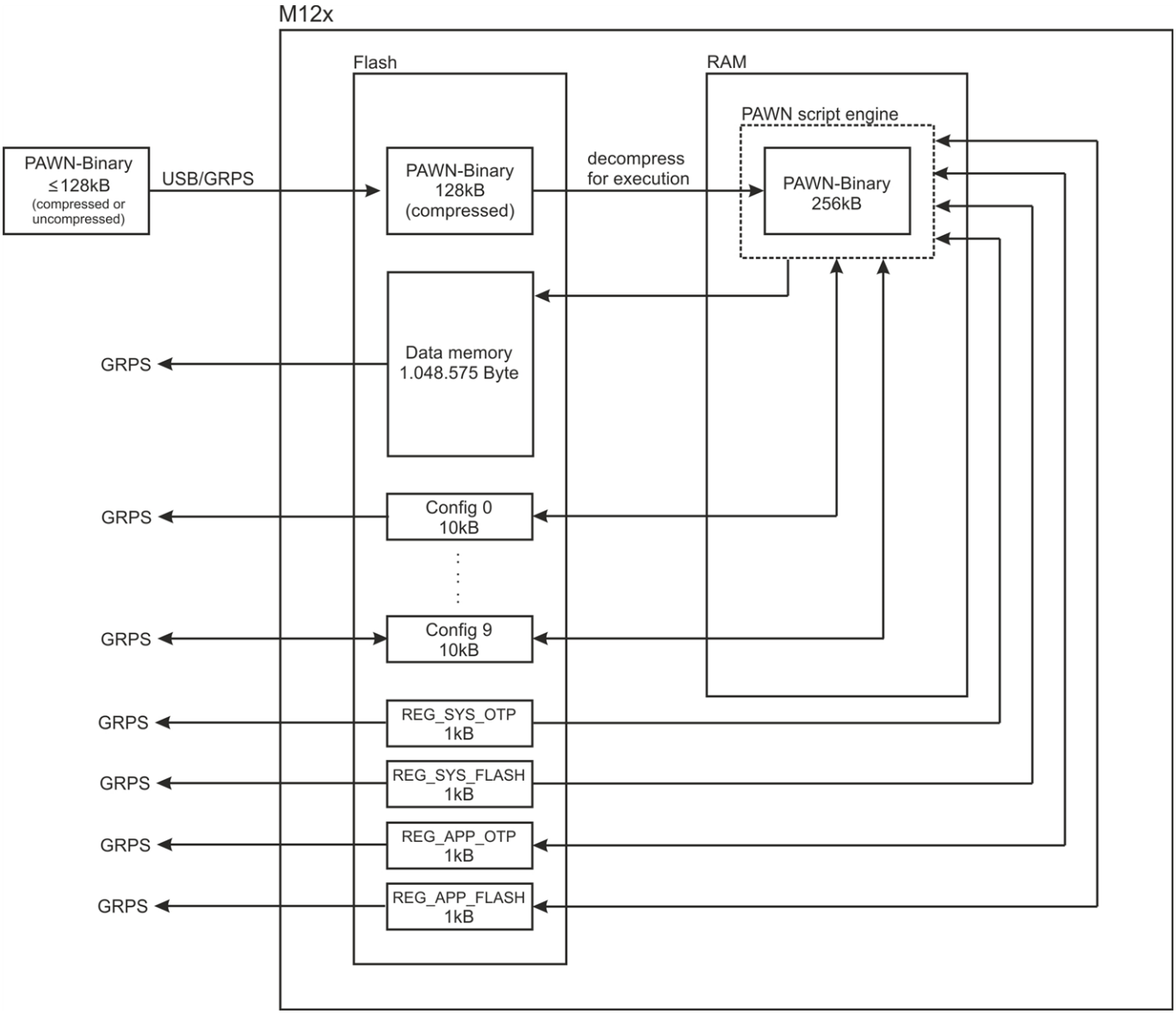


Connector strip (bottom view)

Pin	Type	Title	Description
1	I	EN_{Power}	Pin to activate the DC/DC converter that generates the internal supply voltage V_{AUX} from the voltage connected to V_{IN}
2, 52	I	V_{IN}	Supply voltage (4,5...32VDC) for the M120 module
3	I	EN_{CPU}	Pin to switch on the wireless GSM controller on the M120 module
4, 9, 15, 18, 21, 26, 27, 29, 35, 40, 50		GND	Ground
5	I	Reset	Pin to reset the wireless GSM controller on the M120 module
6, 8, 10, 12, 14, 16, 28		NC	Reserved for extensions No signals must be connected to these pins.
7	O	OD_{LED}	Open drain output to connect a status LED
11, 13			Reserved for extensions No signals must be connected to these pins.
17	I/O	GPIO3	GPIO for exchanging signals between M120 and the surrounding hardware
19	I/O	GPIO2	
20	I/O	GPIO4	
22	I/O	GPIO5	
23	I/O	GPIO1	
24	I/O	V_{AUX}	3,65V voltage output if the M120 module is supplied via V_{IN} Supply voltage input (3,2...4,5V) if no voltage is applied to V_{IN}
25	I	I/O level	2,8...5,5V supply of the level shifters to adjust the signal level of the M120 module to the surrounding hardware
30	O	I2C CLK	Clock line of the I^2C interface
31	I	INT1	Input via which the surrounding hardware can trigger interrupts on the M120 module.
32	I/O	I2C SDA	Data line of the I^2C interface
33	I	INT0	Input via which the surrounding hardware can trigger interrupts on the M120 module.

Pin	Type	Title	Description
34	I	$3V3_{USB}$	USB supply ATTENTION: Only 3,3V are permitted on this pin instead of the usual 5V.
36	I/O	DM	"Data" line of the USB interface
37	O	RTS2	RTS line of UART2
38	I/O	DP	"Data" line of the USB interface
39	I	CTS2	CTS line of UART2
41	O	TxD2	TxD line of UART2
42	O	SPI MOSI	"Master out, slave in" line of the SPI interface
43	I	RxD2	RxD line of UART2
44	I	SPI MISO	"Master in, slave out" line of the SPI interface
45	O	RTS1	RTS line of UART1
46	O	SPI CLK	Clock line of the SPI interface
47	I	CTS1	CTS line of UART1
48	I/O	GPIO6	GPIO for exchanging signals between M120 and the surrounding hardware
49	O	TxD1	TxD line of UART1
51	I	RxD1	RxD line of UART1

7.2.3 Memory organisation



Organisation of the memory of the M120 module

The size of the PAWN binary must not exceed 128kB for the transmission to the rapidM2M module M120 . If necessary, the PAWN binary can be compressed using a compiler instruction (#pragma amxcompress ‹0-3›). The PAWN binary (128kB), 10 configuration blocks (10kB each) and 4 registration memory blocks (1kB each) and measurement 3 x 349.525 Bytedata (3 x 349.525 Byte)are stored in the flash memory of the M120 module. To execute via the PAWN script engine, the PAWN binary is decompressed, if necessary, and copied to the RAM. The maximum size for the decompressed PAWN binary in the RAM is 256kB .

7.2.4 Device labelling

The type plate is located on the top of the module and includes the following specifications:

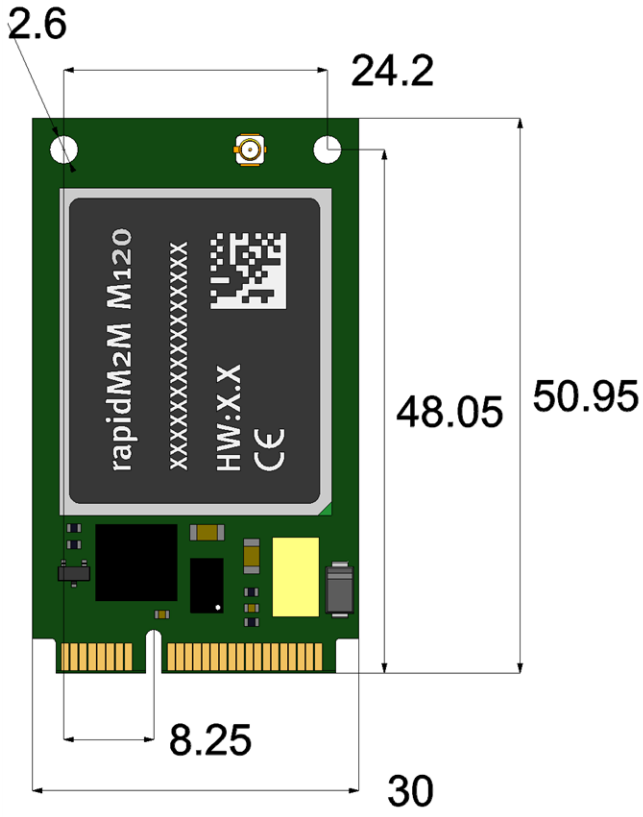
- Type designation
- Serial number
- CE marking
- Hardware revision



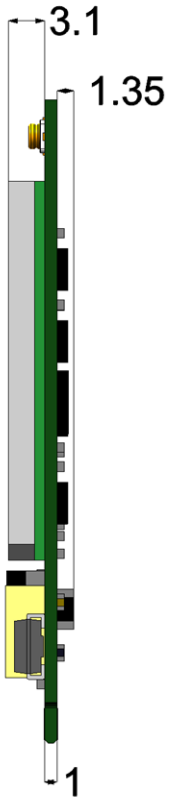
Type plate M120 module

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

7.3 Dimensions

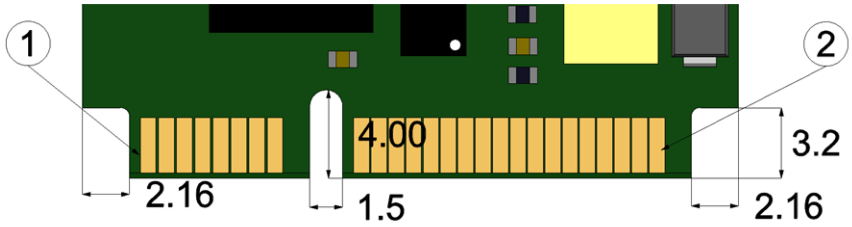


Dimensions: Width and depth (view from above)



Dimensions: Height

7.3.1 Dimensions of the 52pin connector strip (X2)



Detailed overview of the connector strip (from above)

1 Pin 1	2 Pin 51

7.4 Technical details

7.4.1 Technical details about the energy supply

Depending on the voltage available, there are two options to supply the M120 module:

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

Symbol	Parameter	Condition	min.	typ.	max.
V_{IN}	Input voltage	Supply via V_{IN}	4,5V		32V
		Supply via V_{AUX}	---	---	---
I_{IN}	Input current	Supply via V_{IN}		100mA @ 12V DC[1)]	
		Supply via V_{AUX}	---	---	---
$\eta_{DC/DC\ VIN}$	Efficiency of the DC/DC converter	Supply via V_{IN}		94%	
$I_{Q\ DC/DC\ VIN}$	Standby current of the DC/DC converter	Supply via V_{IN}		2,5µA	
V_{AUX}	Output voltage	Supply via V_{IN}		3,65V	
	Input voltage	Supply via V_{AUX}	3,2V		4,5V
I_{AUX}	Output current	Supply via V_{IN}			300mA
	Input current	Supply via V_{AUX}		294mA @ 3,8V DC[1)]	

[1)] The values were determined at a GSM signal strength of -60dBm. Higher current values are possible during an active GSM data transmission. The peak current is dependent on the GSM signal strength, amount of usage data and the transmission speed.

7.4.1.1 Supply via V_{IN}

The internal supply voltage (3,65V) is generated by the DC/DC converter (94% efficiency, 2,5µA standby current) of the M120 module (see "Block diagram" on page 100). The converter must be activated via the "EN_{Power}" pin for this purpose. The DC/DC converter is equipped with a soft start function to ensure the load on the energy source is minimised when activated. It therefore takes 50ms until the internal supply voltage has reached its maximum value. If the M120 module is supplied via V_{IN}, the "V_{AUX}" pin functions as a voltage output for the internal 3,65V supply voltage.

7.4.1.2 Supply via V_{AUX}

The DC/DC converter of the M120 module (see "Block diagram" on page 100) is omitted and the supply voltage is fed directly in to the module via the "V_{AUX}" pin.

7.4.2 EN_{Power} signal

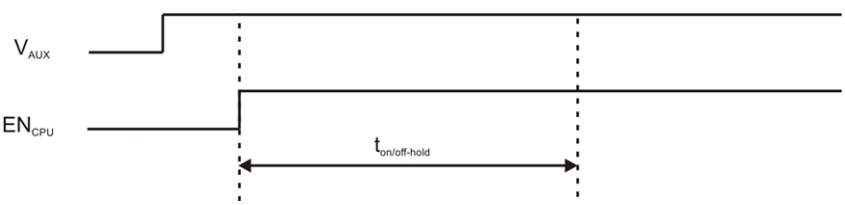
Parameter	min.	typ.	max.
$V_{POWER\ th}$	0,94V	1V	1,06V
$V_{POWER\ hyst}$		40mV	
$t_{soft\ start}$		50ms	

The EN_{Power} signal is only required if the power supply is provided via V_{IN}. The voltage at pin "EN_{Power}" must be above "$V_{POWER\ th}$" (1V) to activate the DC/DC converter for generating the internal supply voltage and the output voltage V_{AUX} (3,65V). The DC/DC converter is equipped with a soft start function to ensure the load on the energy source is minimised when activated. It therefore takes 50ms until the internal supply voltage has reached its maximum value.

7.4.3 EN_{CPU} signal

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

A "high" of at least 1,5sec. ($t_{on/off\text{-}hold}$) on the "EN_{CPU}" pin activates the wireless GSM controller. The "EN_{CPU}" pin should then remain on the "high" level as long as the M120 module is active.



Power ON sequence of the wireless GSM controller

The wireless GSM controller completes an internal reset during the start-up phase. This generally takes 40ms (t_{rst}) and must not be interrupted by an external reset ("reset" pin).

The current firmware of the M120 module does not provide the means to switch off the wireless GSM controller.

7.4.4 Reset signal

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

To initiate a reset, the "reset" pin must be set to "low" via an open collector or an open drain for at least 200µs . The "reset" pin should only be used as an emergency reset, as the wireless GSM controller independently completes a reset during the boot process.

The wireless GSM controller remains in the reset state as long as the "reset" pin remains on "low".

The wireless GSM controller is in the "OFF" state following a reset, which means that it has to be restarted via the EN_{CPU} signal (see "ENCPU signal" on page 106). The "EN_{CPU}" pin can remain "high" during the reset, which causes the wireless GSM controller to restart immediately after it has detected the signal on the "EN_{CPU}" pin (min. 1,5sec.).

7.4.5 Status LED

The rapidM2M module M120 is equipped with an open drain output for connecting a status LED ("OD_{LED}" pin) that lights up for approx. 1sec. during the boot process.

A LED and a resistor can be directly connected between V_{AUX} and the open drain output OD_{LED}.

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

7.4.6 Technical details about the level shifters

> ***Note:*** *The bi-directional level shifters with automatic direction recognition, that are realised in an "open drain" design, are not constructed to drive larger loads.*

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

The module is equipped with level shifters for all of the I/O signals so that the signal level of the M120 module can be adjusted to the surrounding hardware. To complete an adjustment, an additional supply voltage of 2,8...5,5V must be provided for the level shifters by the surrounding hardware. An external voltage or the output voltage from the M120 Module on the "V_{AUX}" pin can be fed in via the "I/O level" pin (see "Block diagram" on page 100).

7.4.7 Serial interfaces

The serial interface levels of the M120 module are dependent on the supply voltage supplied at the "I/O level" pin for the level shifters (see "Block diagram" on page 100).

7.4.7.1 UART

Baud rate	300-115200
Stop bits	1, 2
Parity	N, E, O
Data bits	7, 8

A script example for the use of the UART interface is provided in the chapter tbd.

7.4.7.1.1 Pin description

The UART1 of the M120 module can be accessed via the COM1 connector (J400) of the developer board. The UART2 of the M120 module can be accessed via the COM2 connector (J401) of the developer board. Conversion to the RS232 level is completed by means of a level converter located on the rapidM2M board for developers (see "Block diagram" on page 70).

Pin	Type	Title	Description
37	O	RTS2	RTS line of UART2
39	I	CTS2	CTS line of UART2
41	O	TxD2	TxD line of UART2
43	I	RxD2	RxD line of UART2
45	O	RTS1	RTS line of UART1
47	I	CTS1	CTS line of UART1
49	O	TxD1	TxD line of UART1
51	I	RxD1	RxD line of UART1

7.4.7.2 SPI

Mode	SPI master
Clock polarity/phase	Adjustable
Clock frequency	102kHz...13MHz
Interface	4-wire

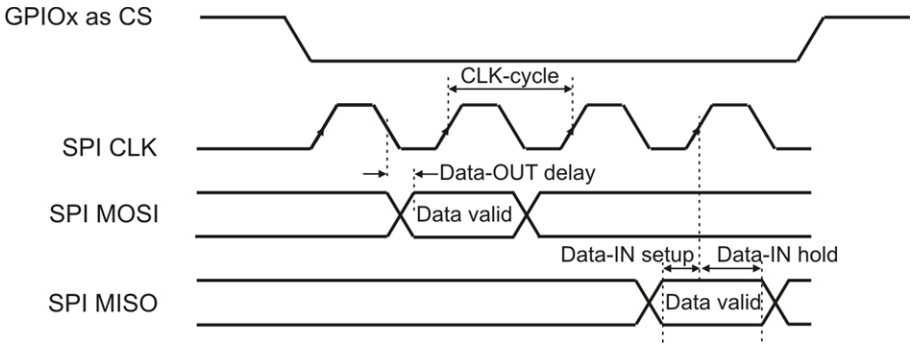
The GPIOs must be used for the CS signal.

A script example for the use of the SPI interface is provided in the chapter tbd.

7.4.7.2.1 Pin description

Pin	Type	Title	Description
17	I/O	GPIO3	GPIOs that can be used for the CS signal
19	I/O	GPIO2	
20	I/O	GPIO4	
22	I/O	GPIO5	
23	I/O	GPIO1	
42	O	SPI MOSI	"Master out, slave in" line of the SPI interface
44	I	SPI MISO	"Master in, slave out" line of the SPI interface
46	O	SPI CLK	Clock line of the SPI interface
48	I/O	GPIO6	GPIO that can be used for the CS signal

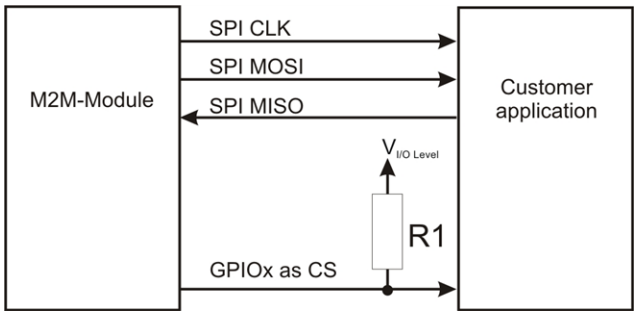
7.4.7.2.2 SPI waveforms



SPI timing diagram (mode 0, master, 4-wire)

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

7.4.7.2.3 Application



Application example

A pull-up resistor R1(min. 10kΩ , typ. 1MΩ) is required to maintain the CS line at a "high" level during the reset state of the M120 module.

7.4.7.3 I²C

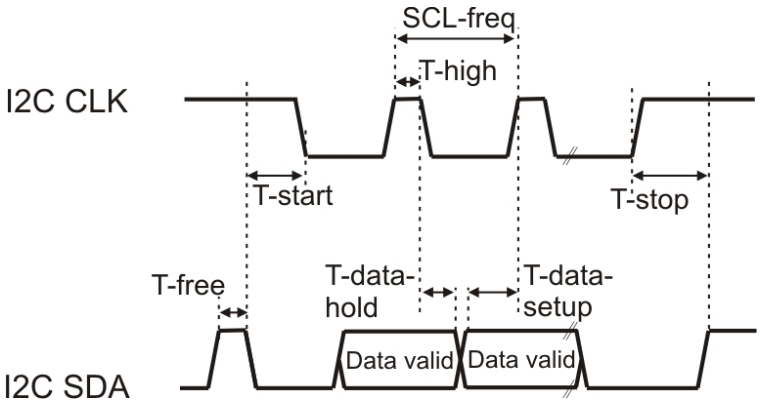
Mode	Master
Clock frequency	100kHz...400kHz

A script example for the use of the I²C interface is provided in the chapter tbd.

7.4.7.3.1 Pin description

Pin	Type	Title	Description
30	O	I2C CLK	Clock line of the I²C interface
32	I/O	I2C SDA	Data line of the I²C interface

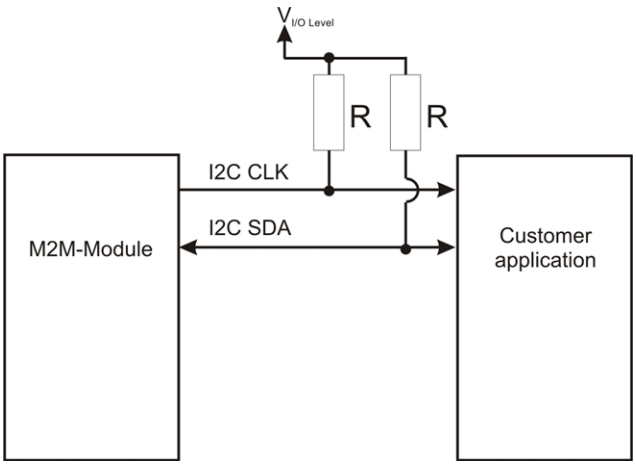
7.4.7.3.2 I²C waveforms



I²C timing diagram (master)

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

7.4.7.3.3 Application



Application example

The CLK and SDA line require a pull-up, that must not be less than 10kΩ .

7.4.8 GPIO

The M120 module comprises 6 GPIOs for exchanging signals with the surrounding hardware. The GPIOs levels are dependent on the supply voltage supplied at the "I/O level" pin for the level shifters (see "Block diagram" on page 100).

A script example for the use of the GPIOs is provided in the chapter tbd.

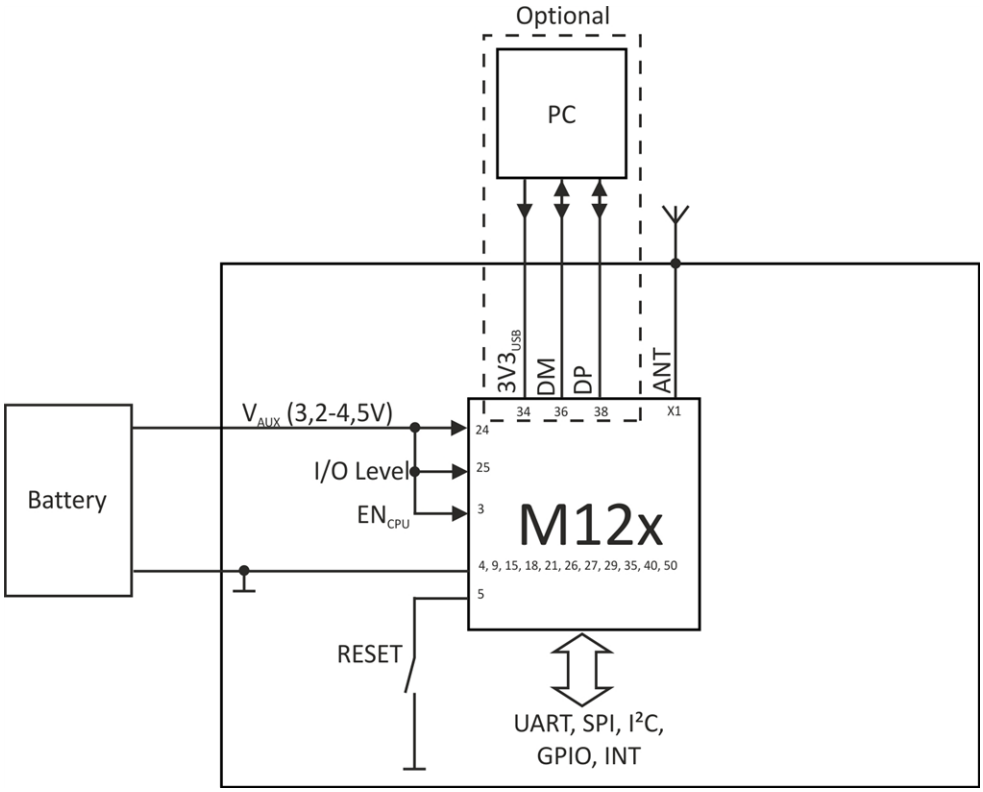
7.4.9 Interrupt inputs

The M120 module comprises 2 interrupt inputs for time-critical applications. These inputs are completely independent of the GPIOs. They can only be used to initiate the execution of the function specified via the "rM2M_IrqInit()" function. It is not possible to read out the state ("low" or "high") at these inputs via the script. The levels of the interrupt inputs are dependent on the supply voltage supplied at the "I/O level" pin for the level shifters (see "Block diagram" on page 100).

A script example for the use of the interrupt inputs is provided in the chapter tbd.

7.5 Typical application

7.5.1 Supply via V_{AUX}

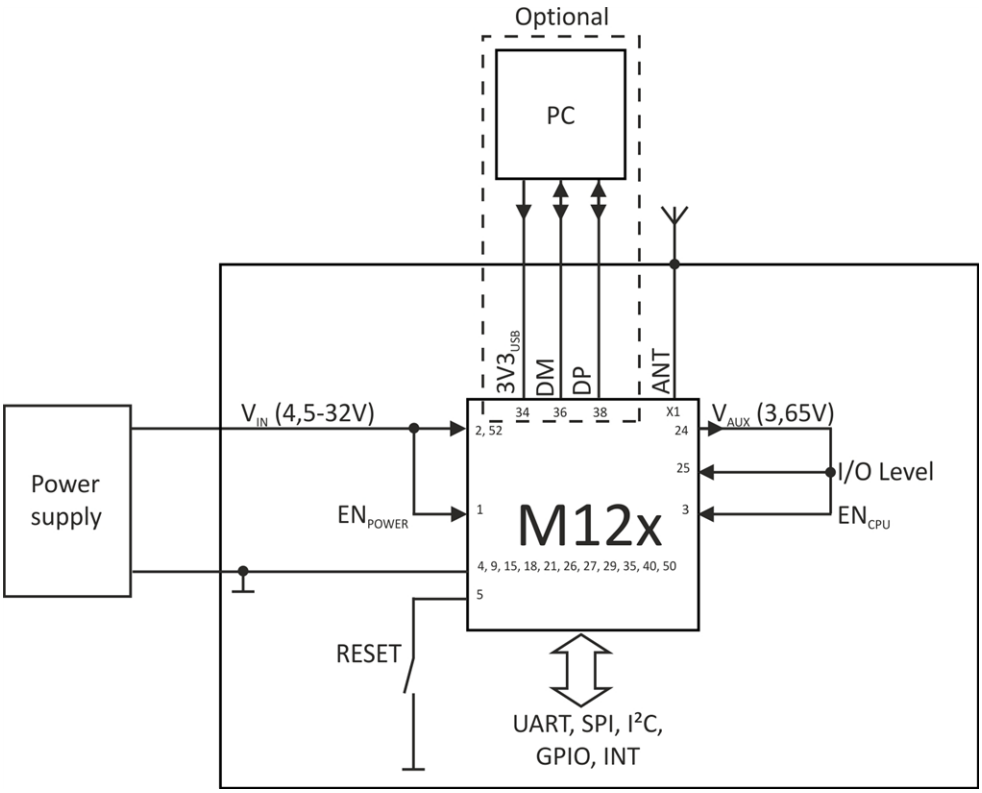


Supply via V_{AUX}

Battery

ICR18650 is a widespread li-ion cell that is perfect for supplying the M120 module. The total impedance must be observed when directly connecting the cell to the V_{AUX} (battery + protective circuit + conducting paths ‹ 150mΩ). Additional measures may be required depending on the operating temperature requirements (voltage delay at low temperatures). The pulse current capacity must be observed when using an alternative cell. Typically, every Li-Ion cell can be discharged with its "C-rating" without any concerns (2Ah --› 2A). The C-rating and the maximum discharge current must be observed for smaller cells (200-500mAh).

7.5.2 Supply via V_{IN}



Supply via V_{IN}

Power supply

Batteries as well as power supplies can be used for the supply. The energy source should supply 6W . DC distribution networks should not be used, as the rapidM2M module M120 does not include circuits to protect against overvoltages. Reverse voltage protection is provided. The DC/DC converter used to generate the internal supply voltage is highly efficient (94% efficiency, 2,5µA standby current).

EN_{Power}

The DC/DC converter for supplying the controller of the M120 module and for generating the 3,65V output voltage (V_{AUX}) is activated via this signal. It must be noted, that when the "EN_{Power}" signal is deactivated, all of the other signals that are applied to the rapidM2M module M120 must be deactivated.

7.5.3 Valid for both applications

USB (optional)

The connection to a PC is established via the USB slave interface (see "USB interface" on page 79).

Antenna

The rapidM2M module M120 is equipped with a GSM/GPRS quad-band modem. The GSM antennas that are used should be selected based on the mobile network frequencies available at the place of use. The optimisation of the antenna for energy-sensitive applications is definitely recommended.

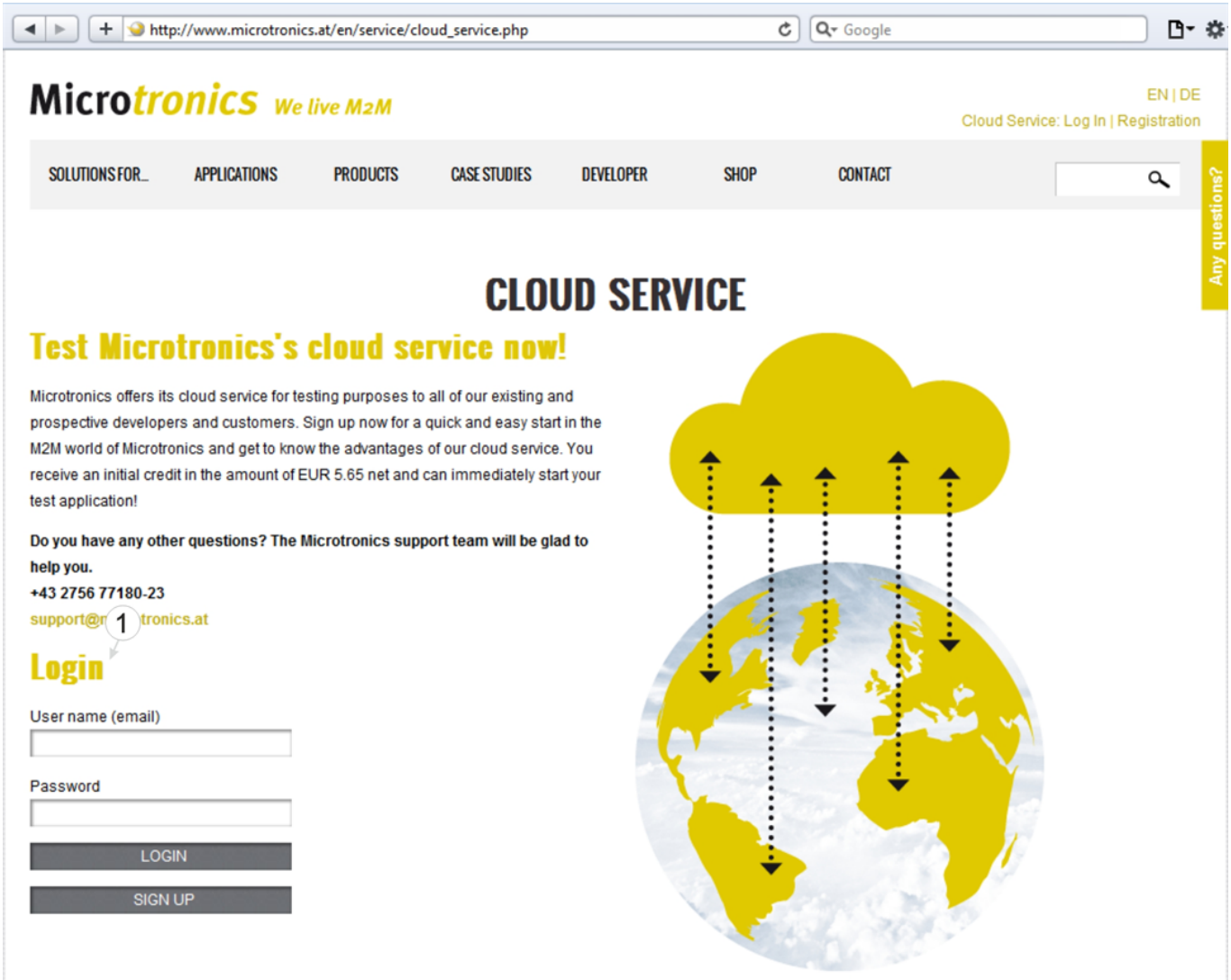
Chapter 8 Cloud server

Note: *All of the screenshots show version 41.1 of the Cloud server. Newer versions may include minor changes to the appearance of the server.*

8.1 New registration and logging in

The Cloud server can be accessed via the Microtronics Engineering GmbH homepage at the following address:

cloud.microtronics.at



Overview of the homepage

1 Access to the Cloud server	

8.1.1 Creating a new account

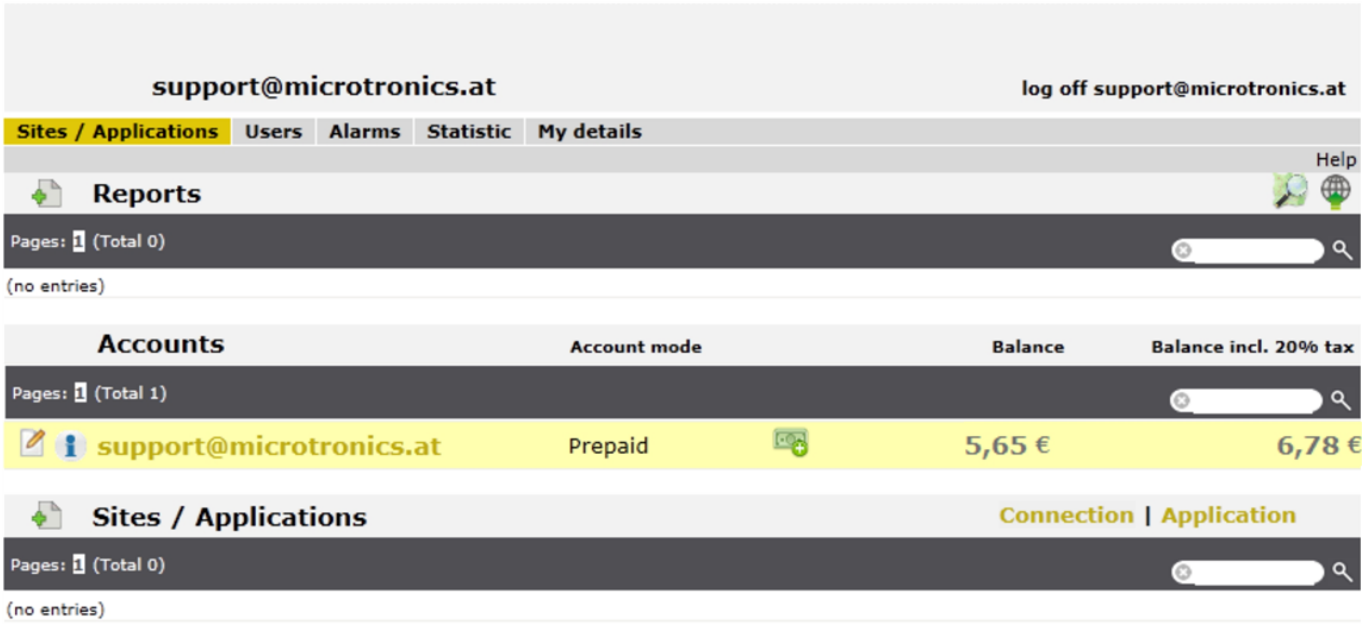
If you do not already have an account on the Cloud server, you can create a free account via the "SIGN UP" button. During this process you are assigned a data volume that is worth 5,65 Euro (excl. VAT). As all of our devices have been optimised to reduce the data volume, the amount of data provided should be enough for approx. one month of average usage. Information on purchasing additional data is provided in the chapter tbd.

The registration form will open once you have clicked on the "SIGN UP" button.

Click on the "REGISTER" button to go to the next step. If all of the mandatory fields have been completed correctly, an e-mail including a link to confirm the registration is sent to the e-mail address specified on the registration form. Click on the link included in the e-mail to complete the registration. You can then log into the account that you have just created on the Cloud server via the Microtronics Engineering GmbH homepage (see "New registration and logging in" on page 115). The e-mail address provided on the registration form must be used as your user name.

8.1.2 Login

Enter your user name (e-mail address) and password in the appropriate fields on the Microtronics Engineering GmbH homepage (see "New registration and logging in" on page 115) and click on "LOGIN". This will take you to the "Sites / Applications " area of the Cloud server.



All of the available sites are displayed in the "Sites / Applications " area.

8.2 Overview



Overview of the Cloud server

1	Name of the active user	3	Tabs to switch between the individual server areas
2	Logout the active user	4	Call up the help menu

8.2.1 Explanation of the symbols



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

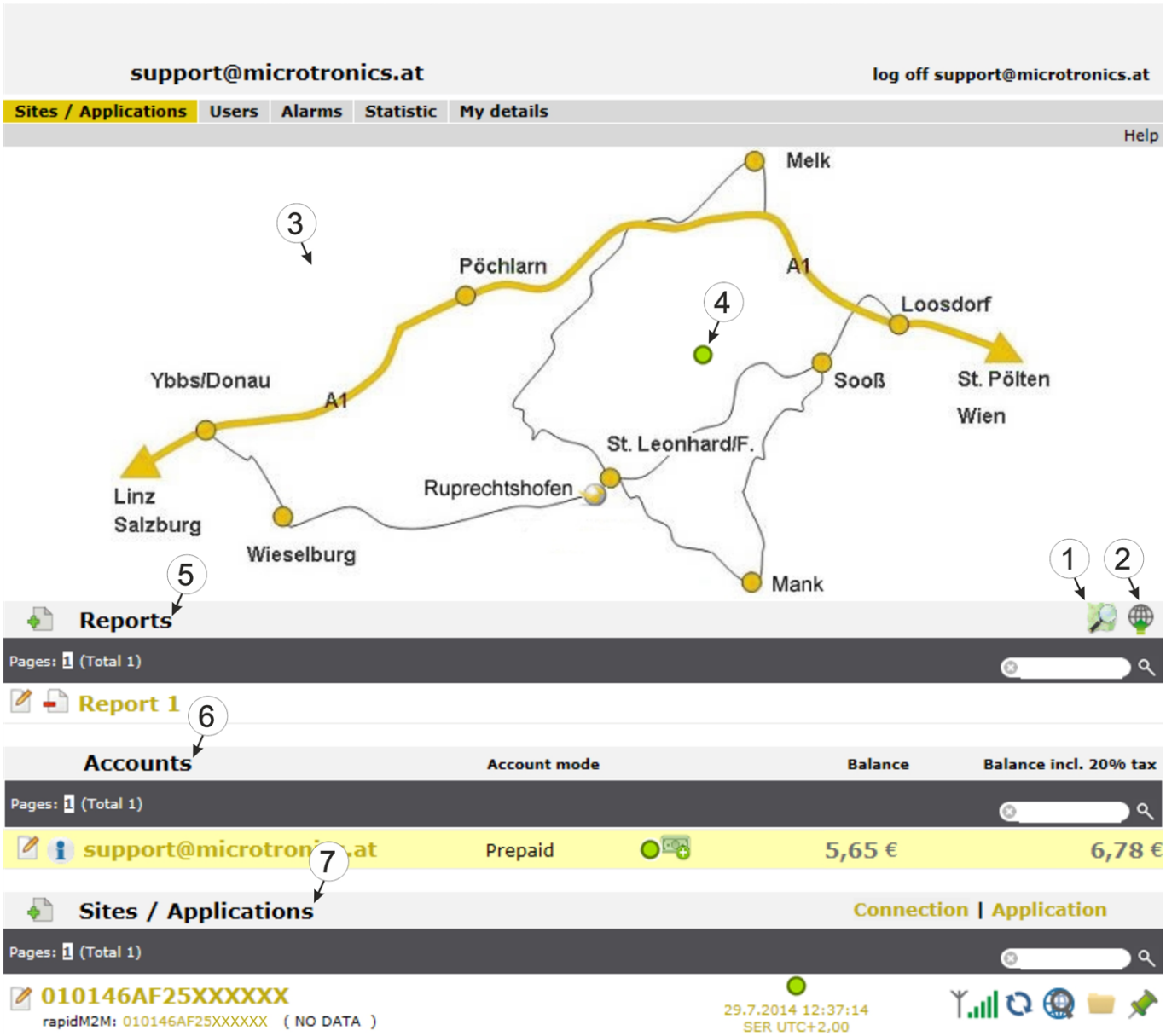


Deletes the adjacent element (reports, sites, users, etc.) from the list.



Calls up the input screen to edit the adjacent element (reports, sites, users, etc.).

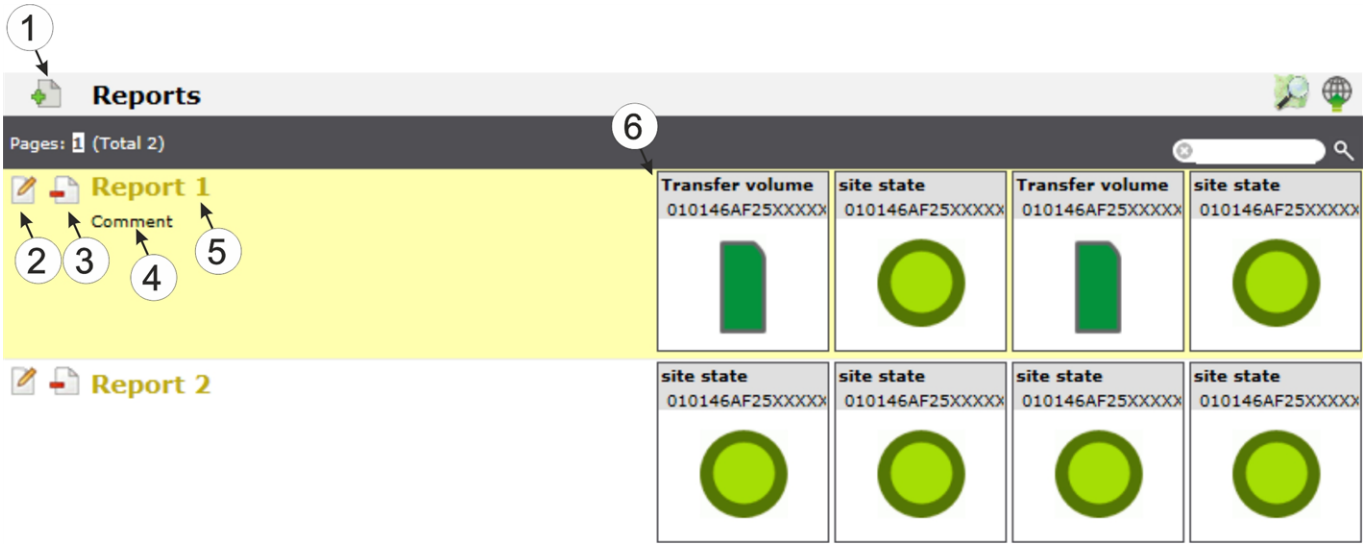
8.3 "Sites" area



Overview of the "Sites" area

1	Symbol via which a OpenStreetMaps map, on which the sites are displayed, can be loaded.
2	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.
3	Current "Map" on which the sites can be placed
4	Symbol that represents a site on the "Map"
5	List of reports
6	List of accounts
7	List of sites / applications

8.3.1 Reports



List of reports

1 Adds a new report(siehe "Basic configuration" on page 119)	4 Comment that can be entered in the basic configuration of the report (see "Basic configuration" on page 119)
2 Opens the input screen for the basic configuration of the report	5 Clicking on the name of the report will open the input screen to configure and select the elements.
3 Deletes the report	6 Display elements for the last measurement value of a measurement channel (max. four per report). The site, channel and type of display element is selected in the basic configuration of the report (see "Basic configuration" on page 119).

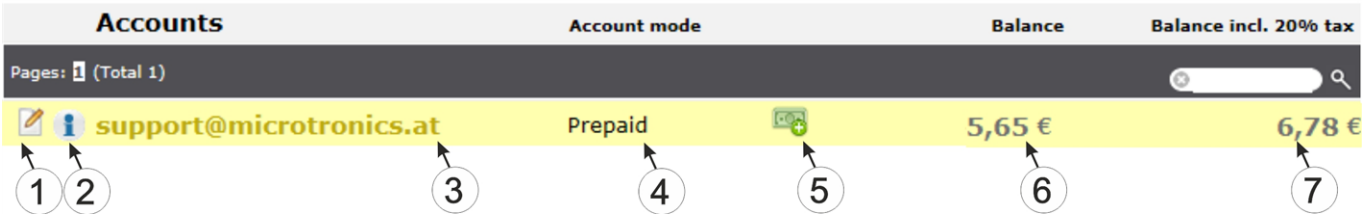
Reports that are highlighted in yellow have been defined as a default report. For this purpose, the type must be set to "Default" in the basic configuration of the report (see tbd). This means that you are immediately directed to this report instead of the "Sites" area once you have logged in via the Microtronics Engineering GmbH homepage. In this case, you can only access the "Sites" area by clicking on the "Menu" (see tbd).

> ***Note:*** *It is, however, possible to set the type to "Default" for several reports. Although in this case, it is not defined which of the reports you will access after logging in.*

8.3.1.1 Basic configuration

This input screen is provided for the basic configuration of the report. Detailed instructions are provided in the myDatanet Server Manual (805002).

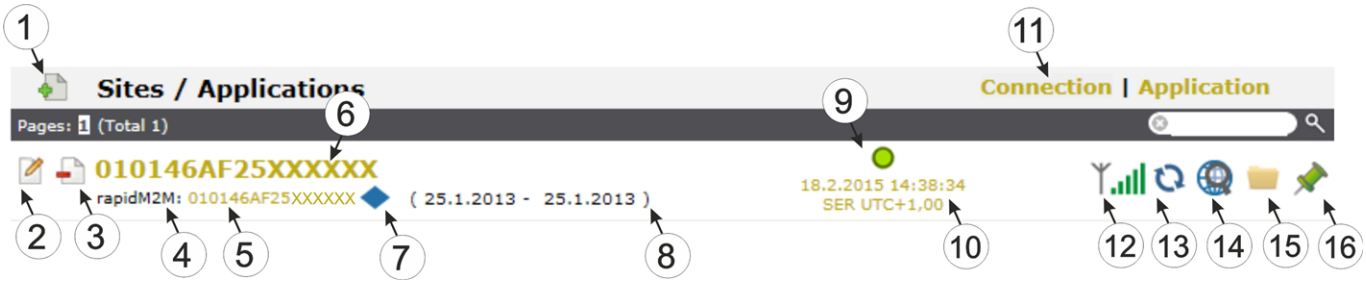
8.3.2 Accounts



List of accounts

1	Opens the input screen to configure the alarm settings for the credit.	5	Opens the input window to load the credit
2	Shows tariff information as a tool tip	6	Current account balance
3	Opens the display of the booking lines	7	Current account balance incl. 20% VAT
4	Type of account		

8.3.3 Sites



List of sites/applications (connection-specific control elements are displayed)

1	Adds a new site/application
2	Opens the input screen for configuring the site (see "Site configuration" on page 189)
3	Deletes the site/application.
4	Type of site
5	Serial number of the device linked to the site. The input screen for configuring the device is opened by clicking on the serial number (see "Device configuration" on page 191).
6	Name of the site/application. The input screen for configuring the site/application is opened by clicking on the name (see "Site configuration" on page 189).
7	The diamond indicates that a firmware update is available for the device. The colour indicates the firmware type (blue for "Released", yellow for "Release Candidate" or brown for "Beta Release").
8	Display of the available data period
9	Status of the measurement instrument. In addition to the status symbol, additional symbols are displayed depending on the current operating state.

Status symbols

🟢	Normal state	⊗	No measurement instrument assigned (only applies to sites)
🔴	Alarm state	○	Measurement instrument does not connect regularly
🟡	Warning state		

Additional symbols

⚠	List contains alarm that has not been acknowledged	↻	Measurement instrument is connected to the server and is transferring data
update	Update in progress...		

10	Time of the last connection. A click opens the connection list.

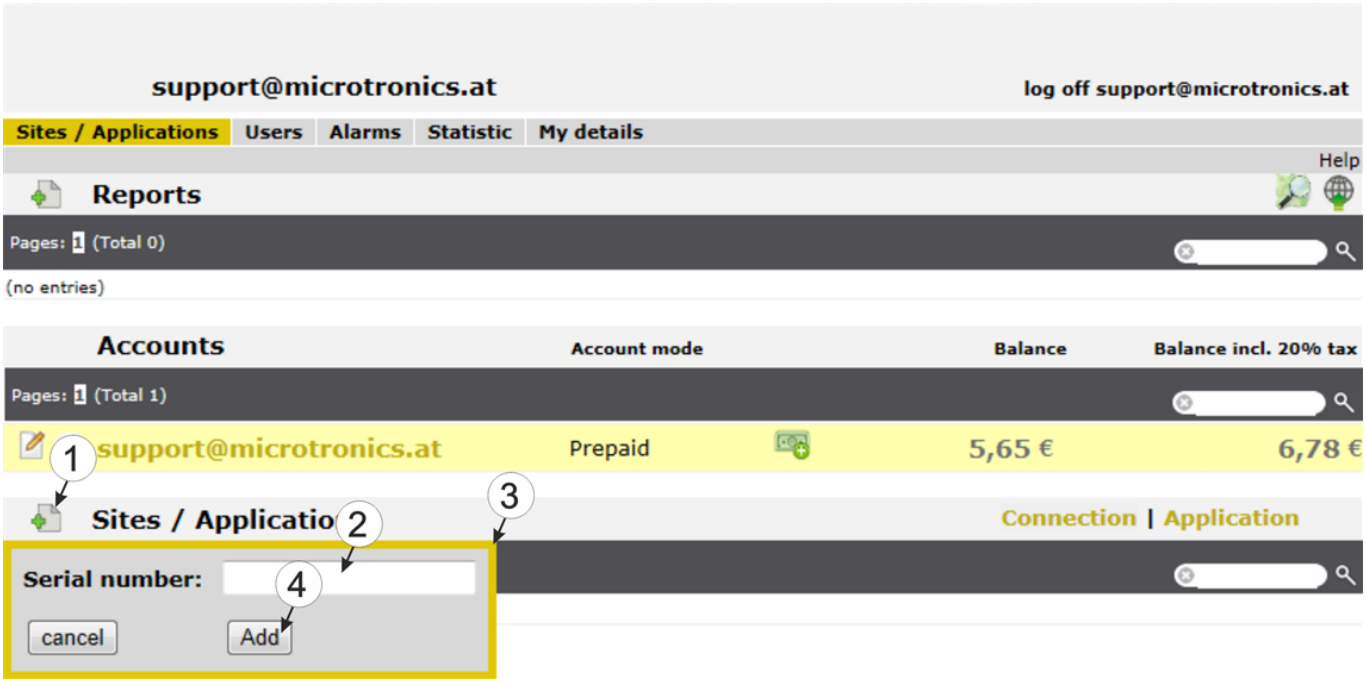
11	Tabs to switch between the display of the connection-specific and application-specific control elements
12	The connection was established via GSM/GPRS. The symbol indicates the signal strength. A click on the symbol displays information about the radio cell used for the last connection and the radio cells taken into account during the last positioning.
13	If the "Interval & Wakeup" connection type is activated on the device, a wakeup SMS can be sent to the device by clicking on this symbol. Upon receipt of such an SMS, the device will immediately establish a connection to the server. ***Important note:*** *Costs are incurred by sending a wakeup SMS.*
14	Indicates the position of the site on the OpenStreetMap map (see "Site position" on page 124). If this symbol is green, GPS coordinates have already been manually assigned to the site. This symbol remains blue as long as the site has not been assigned any GPS coordinates.
15	Opens the input screen for managing the files that are synchronised with the device (see "List of files" on page 125).
16	Opens the input screen for positioning the site on the image file used as a "Map" (see "Positioning the site on the map" on page 127). This symbol is only visible if a corresponding image file has been loaded on to the server



List of sites/applications (the application-specific control elements are displayed:)

1	Tabs to switch between the display of the connection-specific and application-specific control elements	
2	Version number of the application If the application is changed in comparison to the application template for the version number, the version number is highlighted in red and "- changed" is added.	
3	The diamond indicates that an update is available for the application. The new version can be taken over by clicking on the diamond.	
4	Indicates the synchronisation status between the server and device	
		Not all of the settings (configuration, scripts, files, etc.) have been synchronised with the device.
		All of the settings (configuration, scripts, files, etc.) have been successfully synchronised with the device.
		The site/application is not assigned to a device.

8.3.3.1 Creating a new site

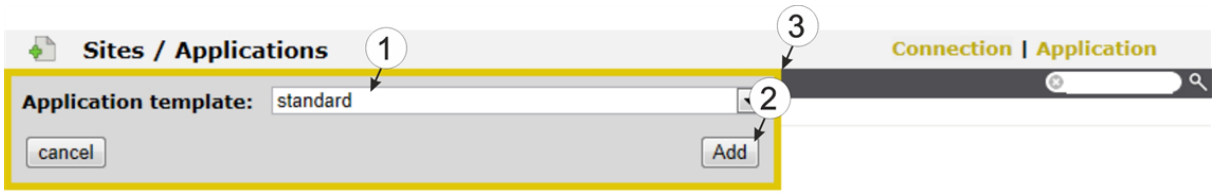


Creating the site

1	"Add new site/application" symbol	3	Input window for creating a new site
2	Field for entering the serial number	4	"Add" button

1. Click on the "Add new site" symbol. The input window for creating a new site will then open.
2. Enter the serial number of your device in the appropriate field. The serial number is on the type plate of the device (see "Device labelling" on page 104)
3. Click on the "Add" button.

4. If desired, select an application template from the list. The new site is then configured according to the template (incl. script). A description of the application templates available for selection is provided in the chapter "Application templates of the Cloud server" on page 136. A site with the default configuration is created if "Standard" is selected.



Selecting the application template

1	Dropdown list of available application templates	3	Input window for selecting the application template
2	"Add" button		

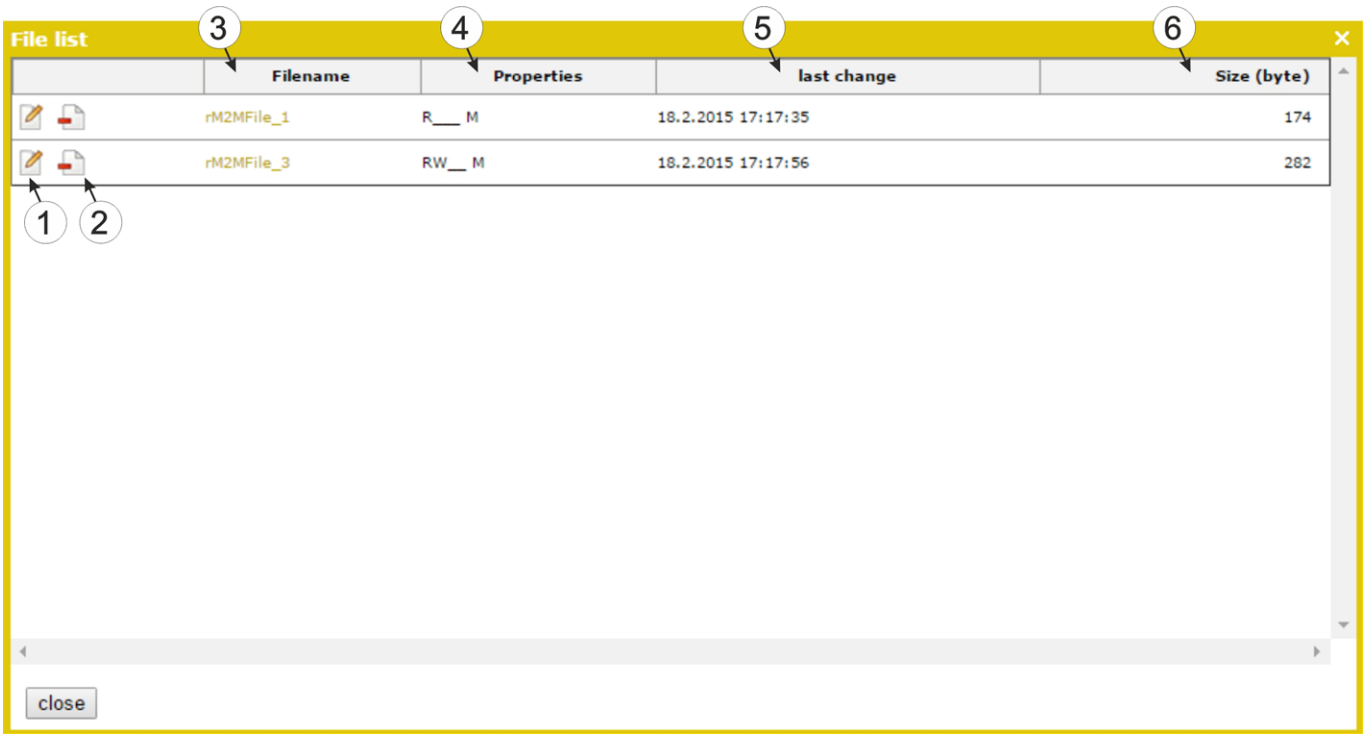
Note: *Detailed information on the application templates is provided in myDatanet Server Manual (805002).*

5. Click on the "Add" button to complete the process.

8.3.3.2 Site position

The position of the instrument is displayed in this window. The frequency at which the instrument must provide its position data can be set with the position cycle in the site settings. Detailed instructions are provided in the myDatanet Server Manual (805002).

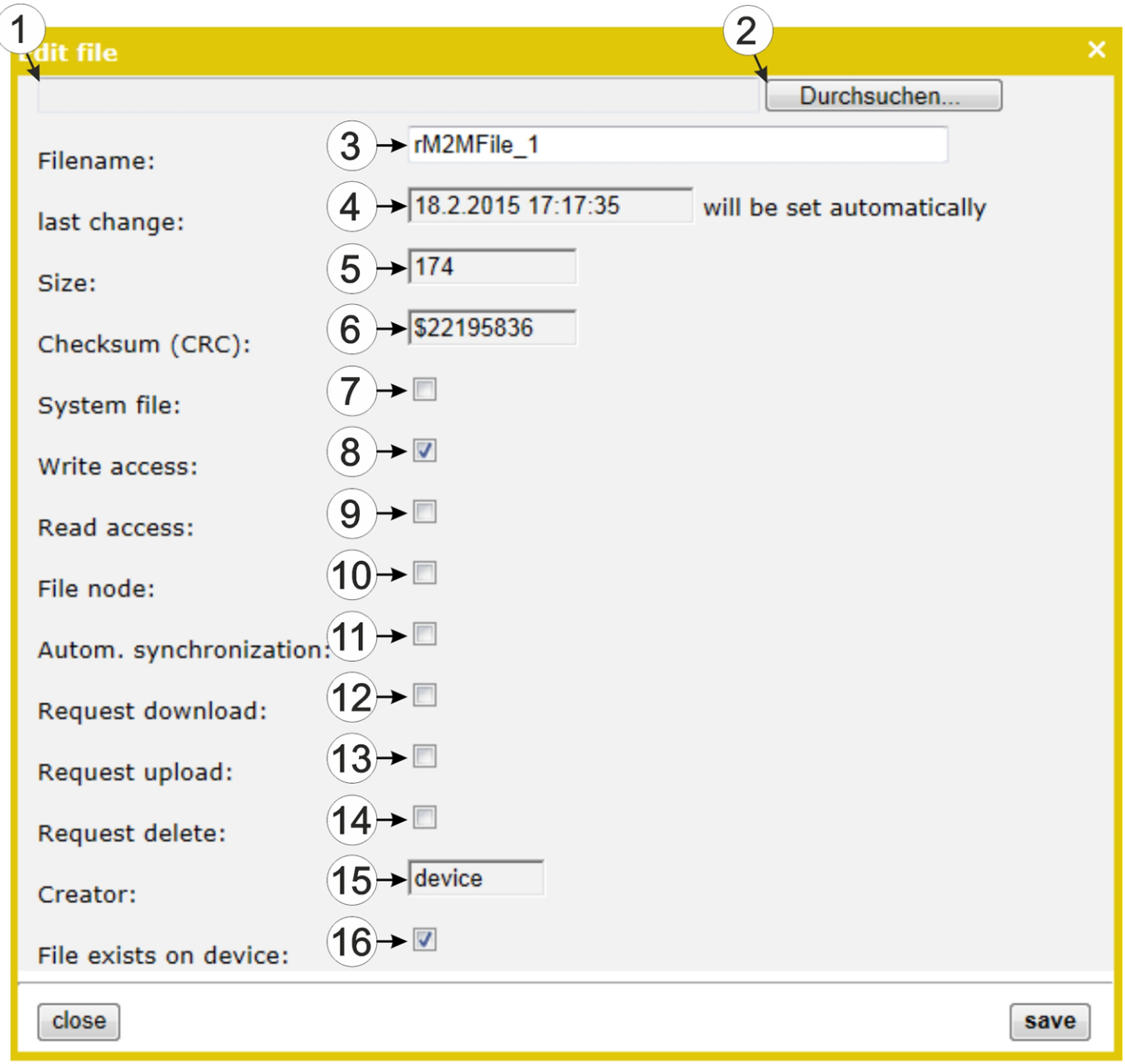
8.3.3.3 List of files



Input screen for managing the files

1	Opens the input screen for editing the file (see "Editing the file" on page 126)
2	Deletes the file from the server. This does not have an effect on the file on the device. The file may be loaded from the device again, depending on the synchronisation settings (automatic/manual). If the file should be deleted from the device, the "Request delete" checkbox in the input screen for editing the file (see "Editing the file" on page 126) should be checked.
3	File name. If the file already contains data (size greater than 0), the file can be loaded from the server by clicking on the name.
4	File properties. These can be edited via the input screen for editing the file (see "Editing the file" on page 126). R ... Read - file can be read from the device. W ... Write - file can be written to the device. N ... Node - file nodes (required to be able to send a file to the device) A/M ... automatic/manual synchronisation
5	Date/time of the last change
6	File size in byte

8.3.3.3.1 Editing the file



Input screen for editing the file

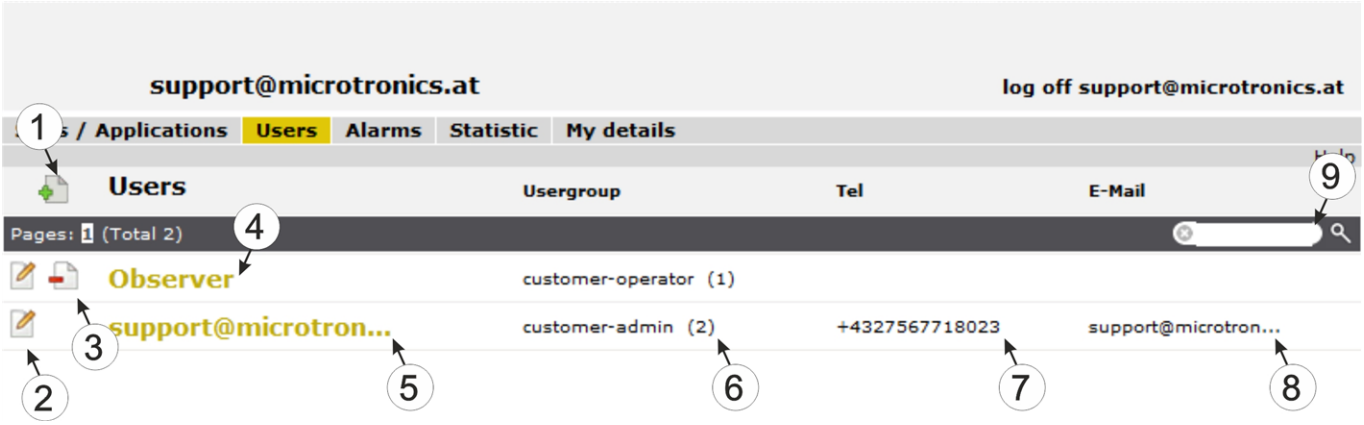
1	File path for uploading the file (only visible as long as the input screen for editing the file is open)
2	Opens the dialogue to select the file to be uploaded
3	File name that is used for the file transfer
4	Date/time of the last change
5	File size in byte
6	Checksum (CRC) that is automatically calculated by the system

7	Indicates whether it is a system file (read only)
8	Option to determine whether the file can be read from the device (selected: read-only access granted)
9	Option to determine whether the file can be saved on the device (selected: write access granted)
10	Option to determine whether it is a file or a file node (selected: file nodes)
11	Option to determine whether the file should automatically be synchronised between the server and the device if any changes are made (selected: automatic synchronisation)
12	The file is loaded to the device during the next connection. This option is only available if automatic synchronisation is not active
13	The file is transferred from the device to the server during the next connection. This option is only available if automatic synchronisation is not active
14	The file is deleted from the device during the next connection
15	Indicates whether the file was created by the device or server
16	Reserved for extensions

8.3.3.4 Positioning the site on the map

The site can be positioned on the image file used as a "Map" via this input screen. Detailed instructions are provided in the myDatanet Server Manual (805002).

8.4 "User" area



Overview of the "User" area

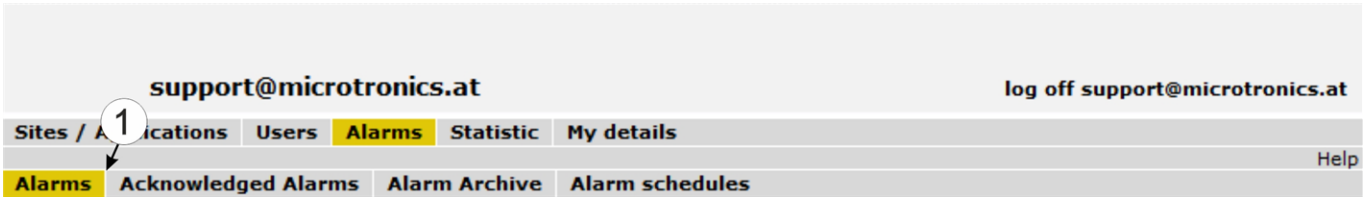
1	Add a new user (see "Creating users" on page 128)	6	Indicates the user group that the user is assigned to
2	Opens the input screen for editing the user data	7	User's telephone number. This number is used for alarm SMS and phone calls.
3	Deletes a user The main user of the Cloud server account cannot be deleted.	8	User's e-mail address. The alarm e-mails are sent to this address.
4	User with less user rights created by the main user of the Cloud server account. Indicated by the "Customer operator (1)" user group. The input screen for editing the user data is opened by clicking on the user name.	9	Search field to filter the user list. Searches for user names, telephone numbers as well as e-mail address are supported.
5	The main user of the Cloud server account is indicated by the "Customer admin (2)" user group The input screen for editing the user data is opened by clicking on the user name.		

The main user of the Cloud server can create additional users with less user rights who will have read-only access to most of the functions provided on the Cloud server. An overview of the rights for individual user groups is provided in tbd.

8.4.1 Creating users

The user data is entered via this input screen. Detailed instructions are provided in the myDatanet Server Manual (805002).

8.5 "Alarms" area



Overview of the "Alarms" area

1	Tabs to switch between the individual sub-sections of the "Alarms" area

The "Alarms" area is split into the following sub-sections:

Alarms
Unacknowledged alarms/warnings

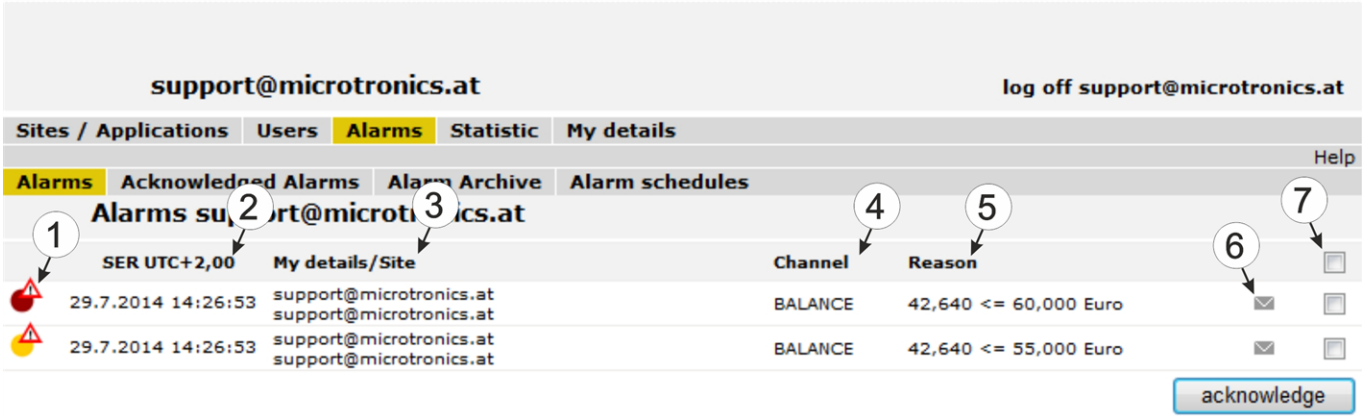
Acknowledged alarms
Alarms/warnings that have already been acknowledged but are still active. Depending on the configuration, the alarms/warnings are acknowledged automatically once all of the notifications have been sent or must be acknowledged manually by the user.

Alarm archive
Alarms/warnings that have already been acknowledged and are no longer active. It is also indicated who acknowledged the alarm/warning.

Alarm schedules
Provided to configure the sources, schedule, recipients and type (e-mail, SMS, phone call) of the notification (see tbd).

> ***Important note:*** *Please note that sending an SMS or making a phone call to indicate an alarm will incur further costs.*

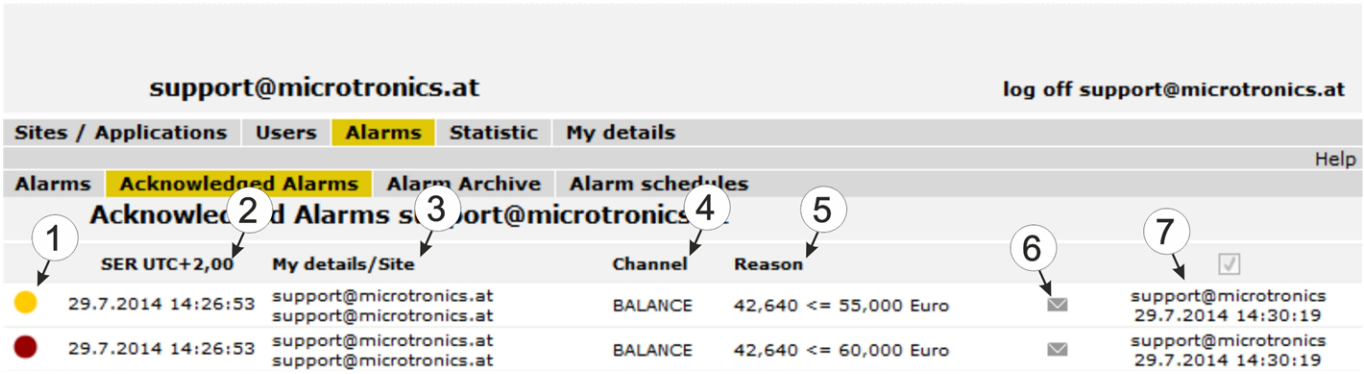
8.5.1 "Alarms" sub-section



"Alarms" sub-section

1	Alarms are indicated by a red status symbol, while warnings are indicated by a yellow status symbol. A superimposed triangle with an explanation mark indicates that an alarm has not been acknowledged ⚠.	5	Reason for the alarm/warning occurring ("triggering value" >= or <= "Level")
2	Time when the alarm/warning occurred (first line) or when it was reset (second line)	6	Details on the sent notifications (see "Notifications" on page 133)
3	User name of the server account (first line) and site or account (second line) where the alarm has occurred	7	Checkbox to select the alarms/warnings that are acknowledged by clicking on the "Acknowledge" button
4	Channel that triggered the alarm/warning		

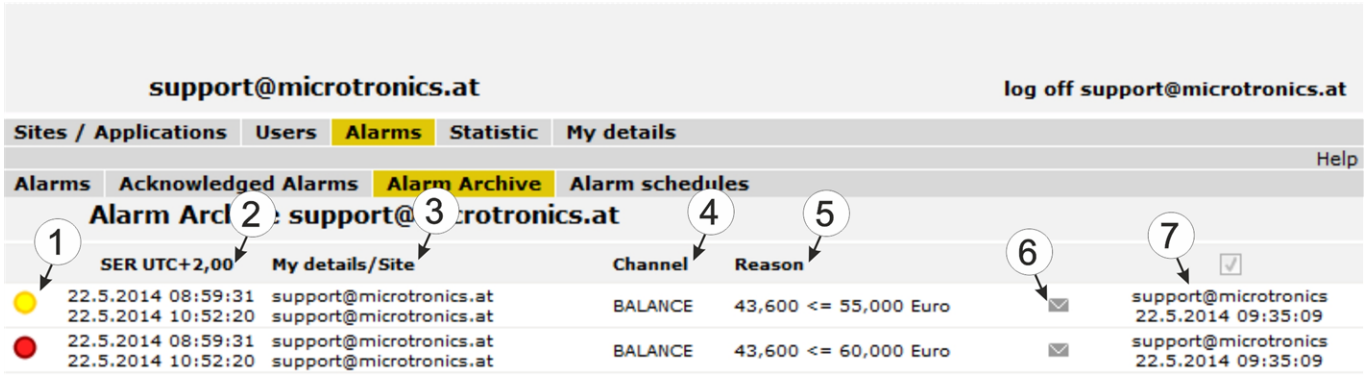
8.5.2 "Acknowledged alarms" sub-section



"Acknowledged alarms" sub-section

1	Alarms are indicated by a red status symbol, while warnings are indicated by a yellow status symbol.	5	Reason for the alarm/warning occurring ("triggering value" >= or <= "Level")
2	Time when the alarm/warning occurred	6	Details on the sent notifications (see "Notifications" on page 133)
3	User name of the server account (first line) and site or account (second line) where the alarm has occurred	7	Name of the user who completed the acknowledgement (first line) and time of the acknowledgement (second line) (only applies to manual acknowledgement)
4	Channel that triggered the alarm/warning		

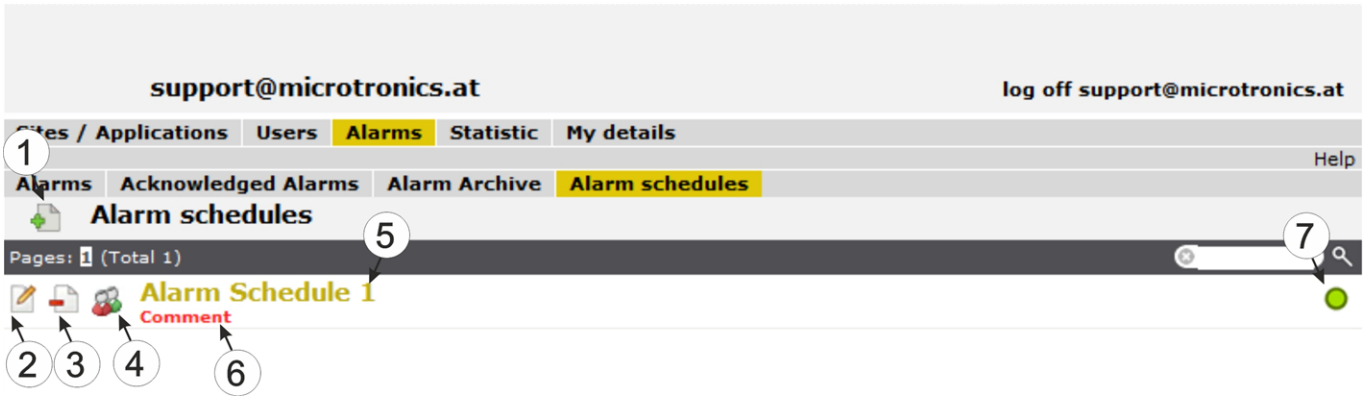
8.5.3 "Alarm archive" sub-section



"Alarm archive" sub-section

1	Alarms are indicated by a red status symbol, while warnings are indicated by a yellow status symbol.	5	Reason for the alarm/warning occurring ("triggering value" ›= or ‹= "Level")
2	Time when the alarm/warning occurred (first line) or when it was reset (second line)	6	Details on the sent notifications (see "Notifications" on page 133)
3	User name of the server account (first line) and site or account (second line) where the alarm has occurred	7	Name of the user who completed the acknowledgement (first line) and time of the acknowledgement (second line) (only applies to manual acknowledgement)
4	Channel that triggered the alarm/warning		

8.5.4 "Alarm schedules" sub-section



"Alarm schedules" sub-section

1	Adds a new alarm schedule	5	Clicking on the name of the alarm schedule will open the input screen to configure the alarm schedule (see"Alarm schedule and comment" on page 133)
2	Opens the input screen for configuring the alarm schedule (see "Alarm schedule and comment" on page 133)	6	Comment that can be entered via the input screen to configure the alarm schedule (see "Alarm schedule and comment" on page 133)
3	Deletes the alarm schedule	7	Status symbol that indicates whether the alarm schedule is active (green: active; grey: inactive)
4	Opens the input screen for configuring the recipient list for the alarm schedule (see "Setup for the alarm by telephone" on page 133)		

8.5.4.1 Alarm schedule and comment

The alarm schedule is configured and a comment can be added to the alarm schedule via this input screen. Detailed instructions are provided in the myDatanet Server Manual (805002).

8.5.4.2 Setup for the alarm by telephone

The alarm by telephone is configured via this input screen. Detailed instructions are provided in the myDatanet Server Manual (805002).

8.5.5 Notifications

This input screen contains information on the sent messages. Detailed instructions are provided in the myDatanet Server Manual (805002).

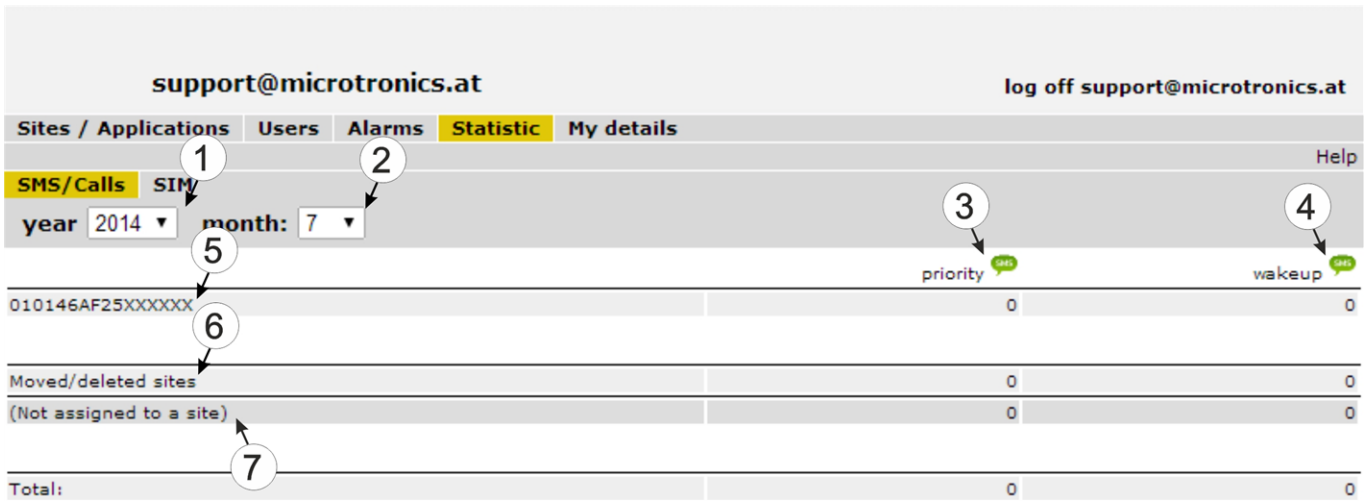
8.6 "Statistics" area



Overview of the "Statistics" area

1	Tabs to switch between the individual sub-sections of the "Statistics" area

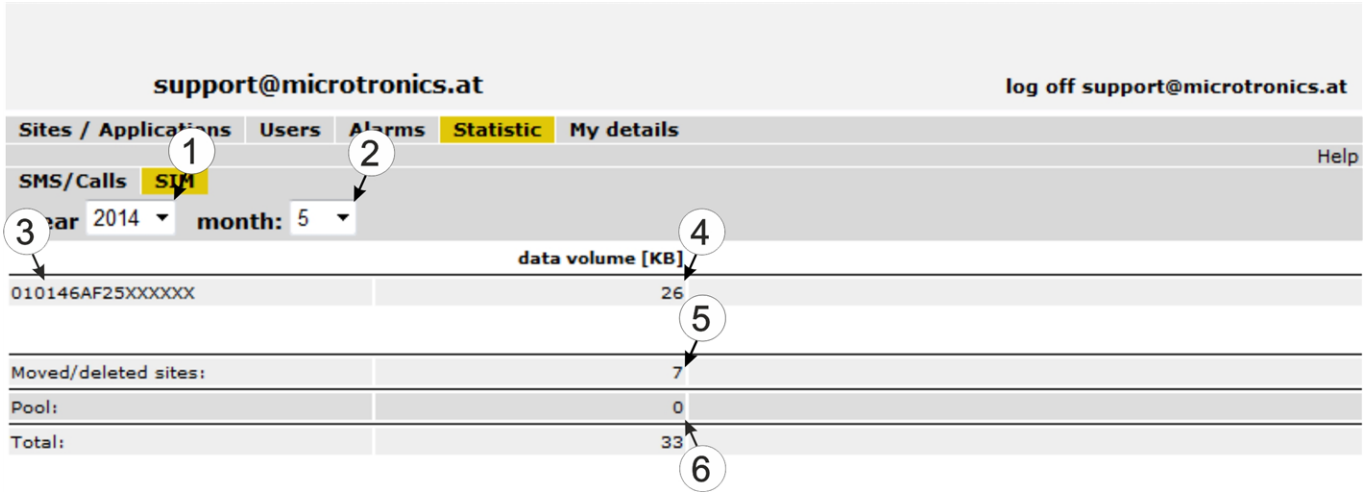
8.6.1 "SMS/voice calls" sub-section



"SMS/voice calls" sub-section

1	The statistics are always generated for a month. The year is initially selected via this drop down list. The month is then selected using the "month" drop down list.
2	Selection of the month for which the statistics should be created
3	Number of high-priority SMS that were sent. These SMS are delivered with a higher priority. Feedback is provided to confirm successful delivery. However, whether the SMS has actually been read cannot be determined by this feedback. The number of SMS sent are listed for each site.
4	Number of wakeup SMS sent. If a device is in wakeup mode ("Interval & wakeup" connection type), it can be placed in ALOHA transmission mode via the server. For this purpose, a SMS is sent from the server to the device. The number of SMS sent are listed for each site.
5	List of active sites
6	List of SMS caused by displaced or deleted sites. This is the number of SMS that were generated up until the sites were displaced/deleted.
7	List of SMS that cannot be directly assigned to a site; e.g. caused by a test alarm being triggered

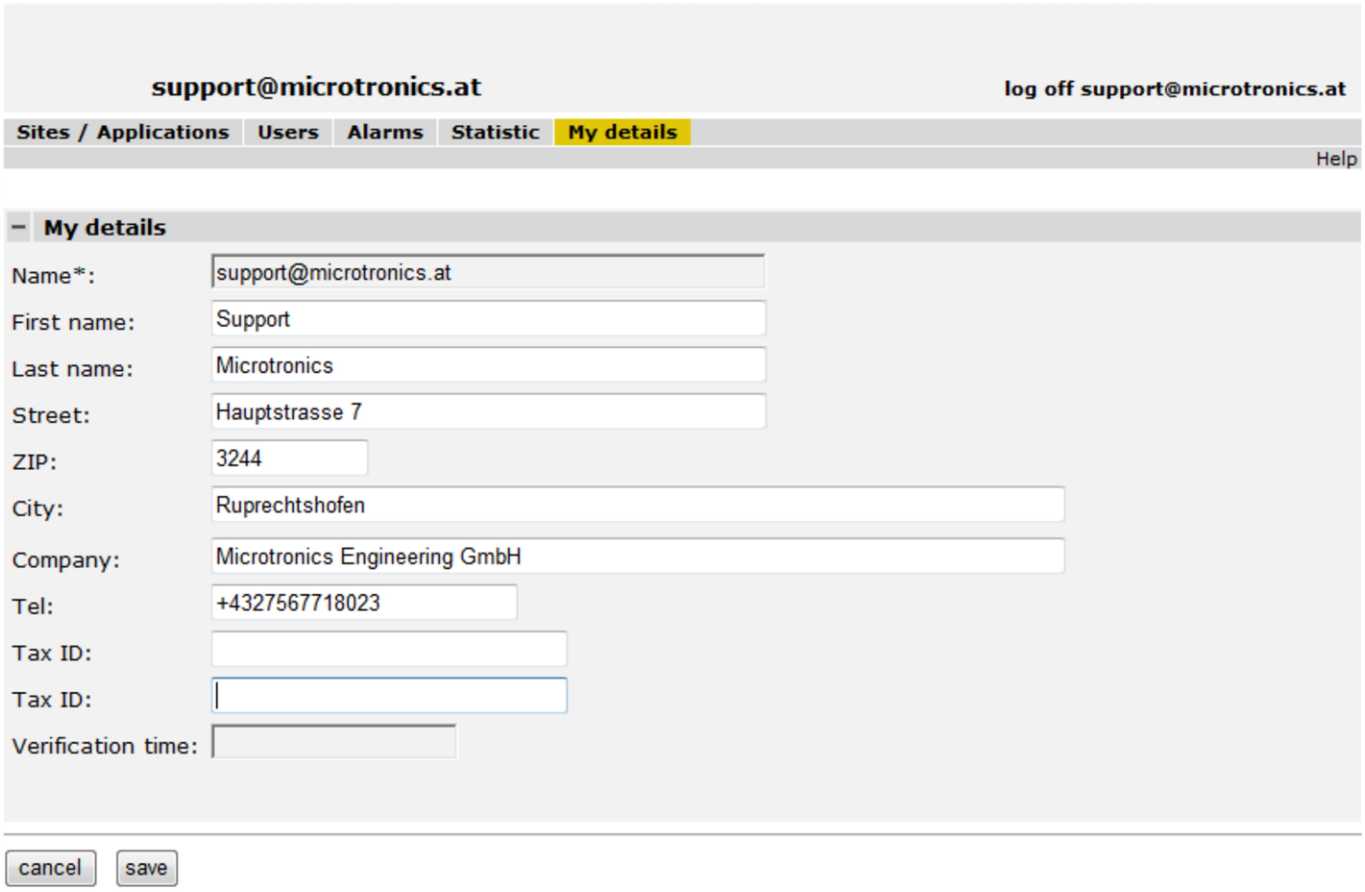
8.6.2 "SIM" sub-section



"SIM" sub-section

1	The statistics are always generated for a month. The year is initially selected via this drop down list. The month is then selected using the "month" drop down list.
2	Selection of the month for which the statistics should be created
3	List of active sites
4	Used data volume listed according to the individual sites
5	Data volume used by displaced or deleted sites. This is the data volume that was used up until the sites were displaced/deleted.
6	Data volume used by devices in the pool. This is the data volume that was used from the time when the device was moved into the pool.

8.7 "My data" area



Overview of the "My data" area

All of the data entered during the registration, except for the user name of the Cloud server account can be edited in this area.

8.8 Application templates of the Cloud server

tbd

Chapter 9 rapidM2M Toolset

Note: *All of the screenshots show version 1.12 of the rapidM2M Toolset . The newer versions may include minor changes to the appearance of the program.*

9.1 General

The rapidM2M Toolset is available at the following website and can be downloaded free of charge:

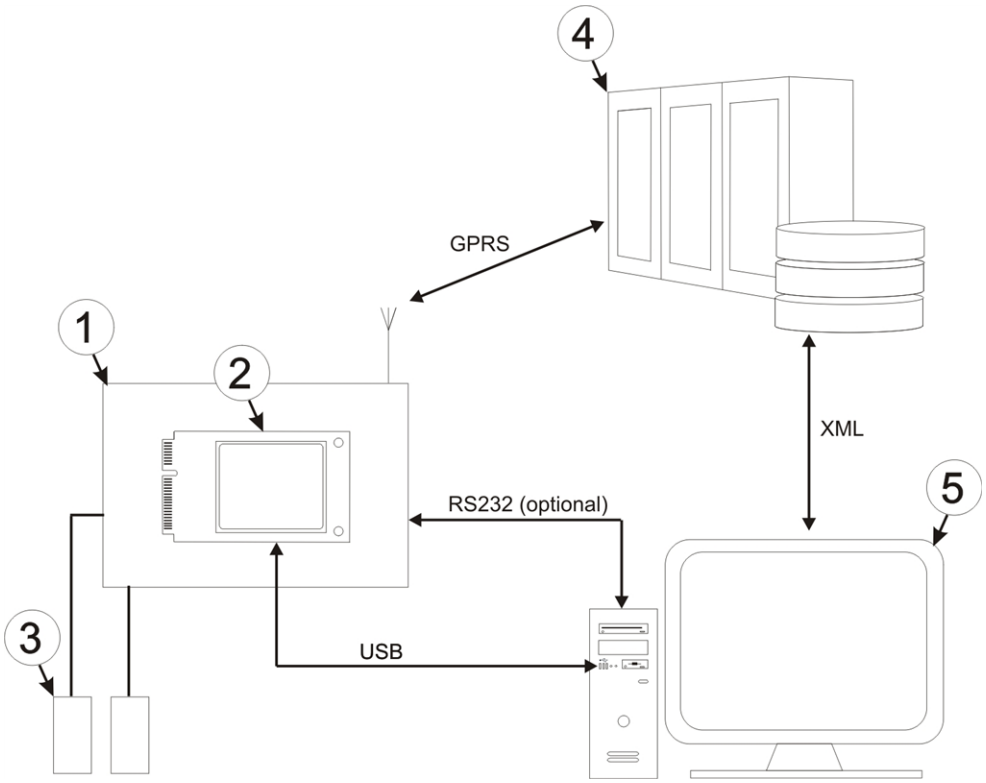
www.microtronics.at/Toolset

It is an IDE that is designed to support the customer with the creation of the applications for the M120 module. This comprises the entire development process, starting with editing the PAWN script (see "Pawn script" on page 193), to compiling the script in byte code and loading the byte code in the M120 module, through to reading out the transferred data from the M120 module to the Cloud server via the XML interface of the Cloud server. The rapidM2M Toolset enables the display of the debug information, that was printed to the standard output within the PAWN scripts by means of the "print" and "printf" functions, and also enables the log file to be loaded from the M120 module. The 10 configuration memory blocks can also be read out of the M120 module or written to the M120 module via the rapidM2M Toolset . For test purposes, the rapidM2M Toolset includes the option to simulate a simple serial sensor and includes a collection of example scripts. Finally, the rapidM2M Toolset can also be used to install firmware updates on the M120 module.

9.2 Prerequisites

Interfaces	1 x USB 1 x RS232 (optional)
Operating system	Win XP Windows Vista Windows 7
Internet connection	Recommended
Required disk space	approx. 75MB

9.3 Functioning principle



Functioning principle

1	Device (customer-specific hardware) including installed M120 module; e.g.rapidM2M board for developers	4	Cloud sever
2	M120 module	5	PC on which the rapidM2M Toolset is installed
3	Sensor that is connected to the device (customer-specific hardware) (optional)		

The rapidM2M Toolset communicates directly with the M120 module via the USB interface. This includes:

- Script download (see ""Script" tab" on page 144)
- Up/download of the configuration memory blocks (see ""Config" tab" on page 146)
- Display of the console outputs (see ""Console" tab" on page 147)
- Reading out the log entries (see ""Log" tab" on page 149)
- Firmware update (see ""FW" tab" on page 151)

The RS232 connection is optional and is only required when the option to simulate a simple serial sensor (see ""COM Data" tab" on page 147) of the rapidM2M Toolset is going to be used.

The rapidM2M Toolset also includes the option to read out the data via the XML interface of the Cloud sever (see ""Server" tab" on page 149). This process can be used to quickly and simply check

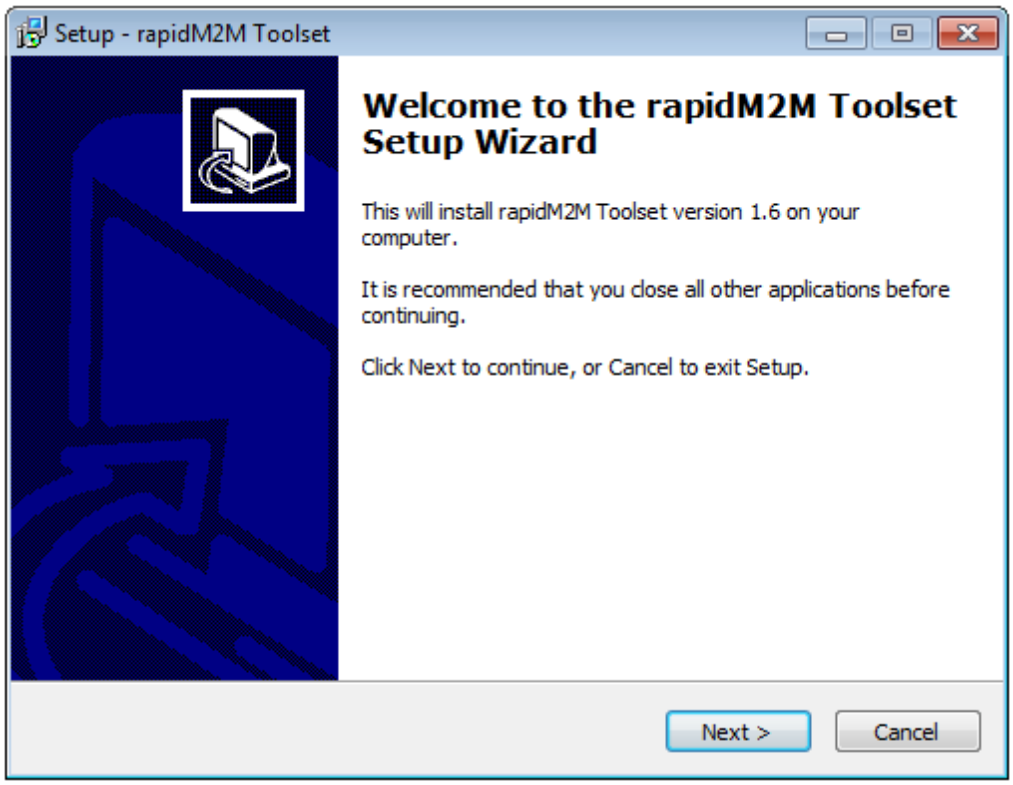
whether the complete process chain, starting with saving the data via the script, through to the transfer per GPRS and storage of the data on the server, works correctly.

9.4 Installation

The following chapter describes the installation process in Windows 7.

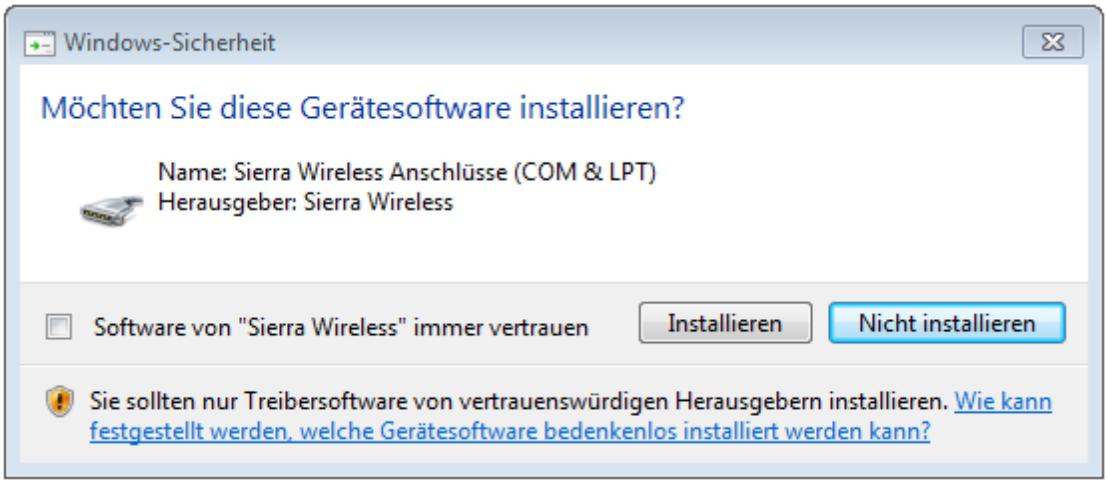
1. Execute the *"InstRapidM2MToolset.exe"* file to start the installation process.

   Note: *Only connect the rapidM2M developer package M120 to your PC once the installation process has completed as the required USB drivers are only installed during this process.*

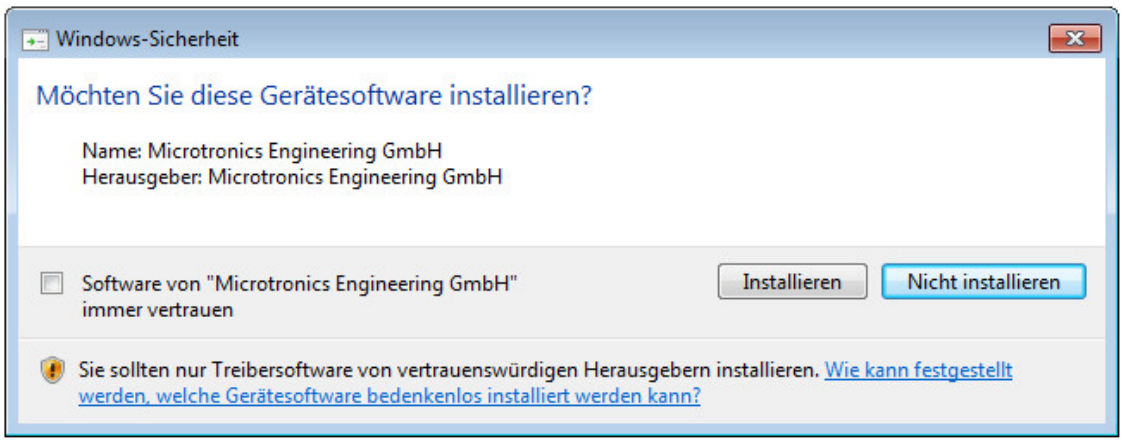


rapidM2M Toolset setup wizard

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

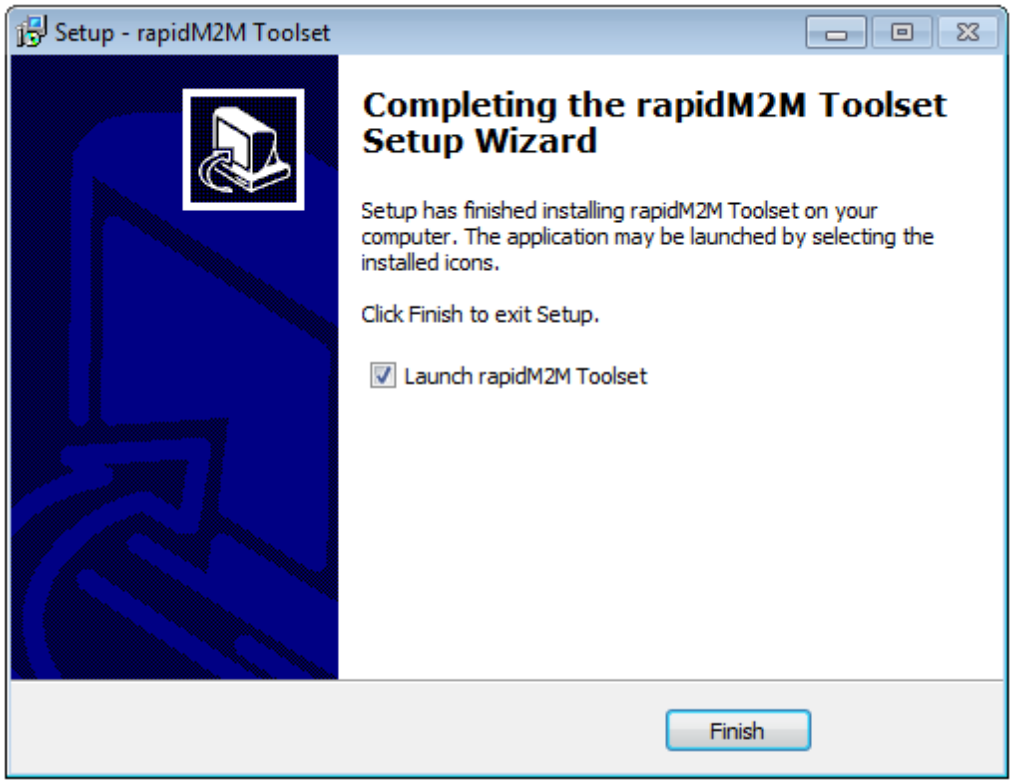


Installing of the USB drivers for the M120 modules



Installing of the USB drivers for the M22x modules

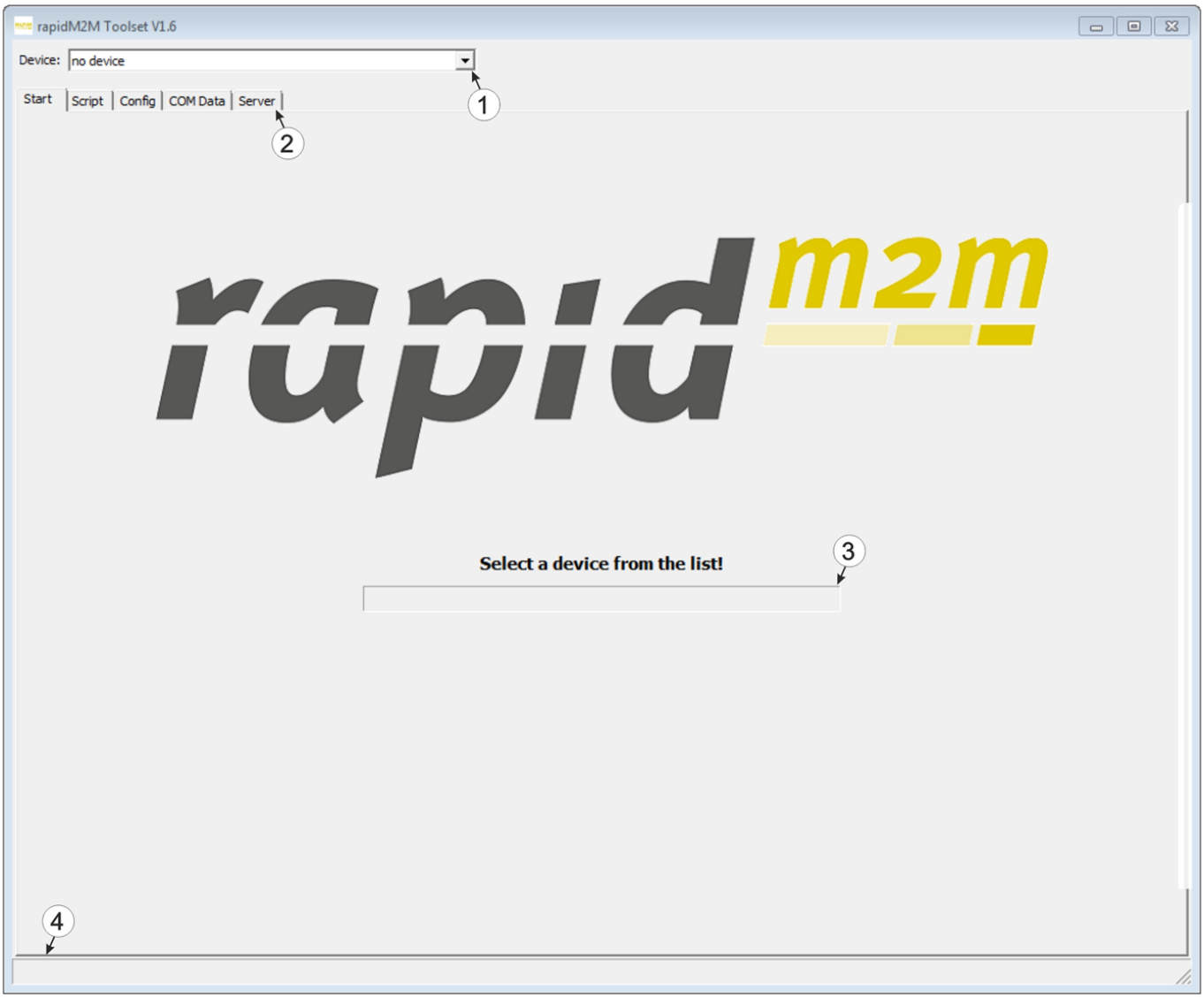
3. Once the following screen is displayed, close the installation process by clicking on the *"Finish"* button.



Complete the setup.

4. The installation of the USB drivers will start after you have clicked on the *"Finish"* button.

9.5 Overview



"Start" tab (no M120 module selected)

1	Selection of the M120 module	3	Progress bar for establishing the USB connection to the M120 module
2	Selection of the tab for the various functions of the rapidM2M Toolset that are even available without the connected module	4	Indication of the communication status on the USB interface to the M120 module

As soon as the USB connection to a M120 module has been established, the tabs for the functions that require a USB connection to the module are displayed and the "Start" tab is hidden. Further information on the module is also displayed in the top right area of the rapidM2M Toolset .

Important note: *Assign the M120 module to a site (see "Creating a new site" on page 123) before accessing it via USB. If this assignment is completed subsequently, all of the settings implemented to date (incl. the script) would be lost as the settings from the site would be adopted. The message "`Warning: No Context available! All settings and script will be lost when device is assigned to a site`" in the status bar for the communication status on the USB interface indicates that no site has been linked to the M120 module.*



Connection to the M120 module established

1	Selection of the M120 module	4	Display of the GSM signal strength (see "GSM signal strength")
2	Selection of the tabs for the various functions of the rapidM2M Toolset	5	Serial number of the connected module
3	Display of the GSM connection status (see "GSM connection status")		

GSM connection status

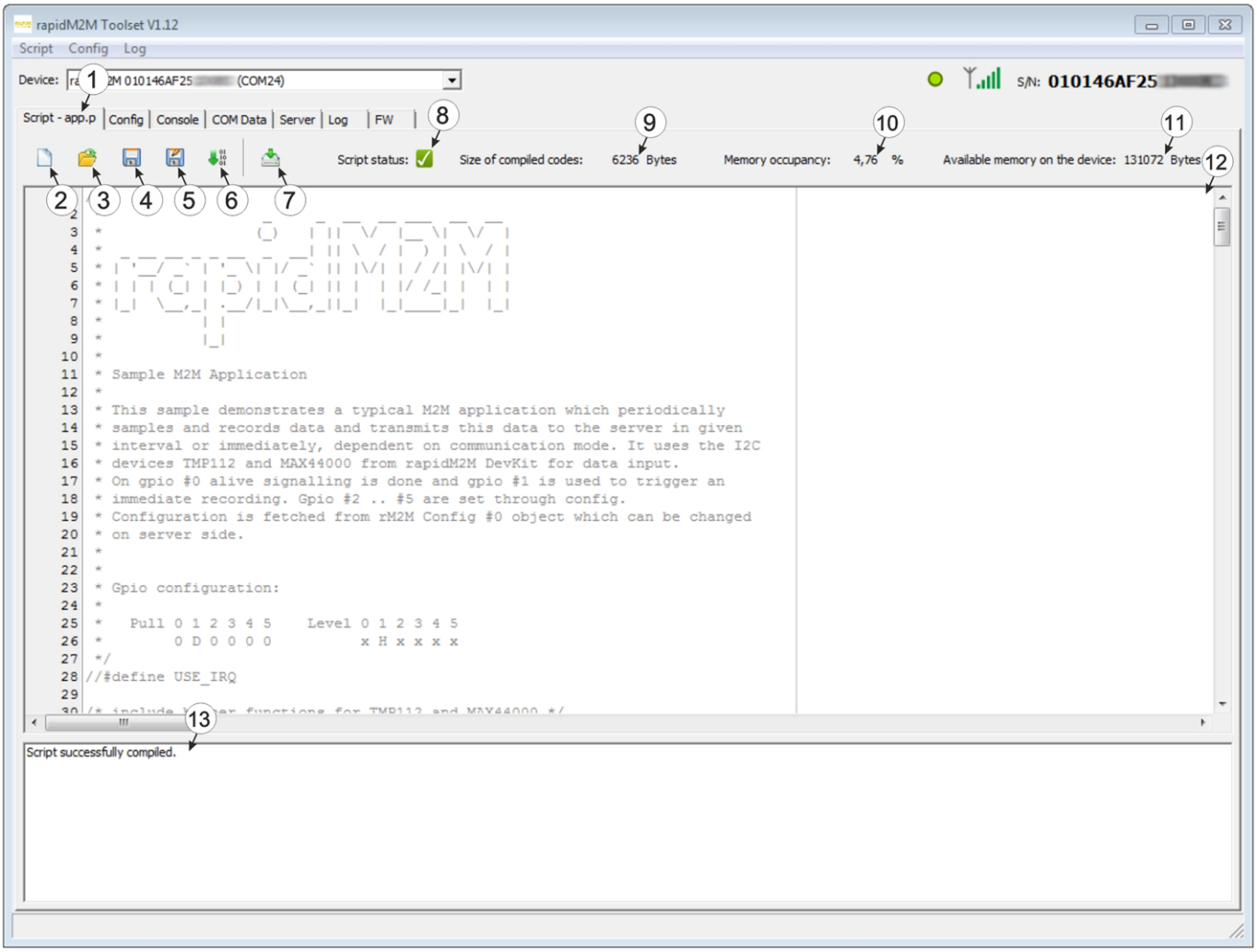
Status symbol	
Grey	Offline
Flashing green	Connection is being established
Green	Connected
Red	Last connection attempt failed

GSM signal strength

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

9.6 "Script" tab

This area is designed to create the script. The tab is also available if there is no USB connection to a M120 module. In this case all of the functions, except for downloading the compiled script and display of the memory sizes ("Size of the compiled codes", "Memory occupancy" and "Available memory on the device") are available. The last script that was opened is automatically loaded when the program starts. If the script that has just been opened has been changed by another program, a message is issued providing the option to load the changed script into the script editor.



"Script" tab

1	File name of the opened script. If the script was changed but has not been saved yet, this is indicated by "*" after the file name.
2	Creates a new empty script
3	Opens a saved script
4	Saves the script that is currently open
5	Opens the dialogue to save the script under a different file name

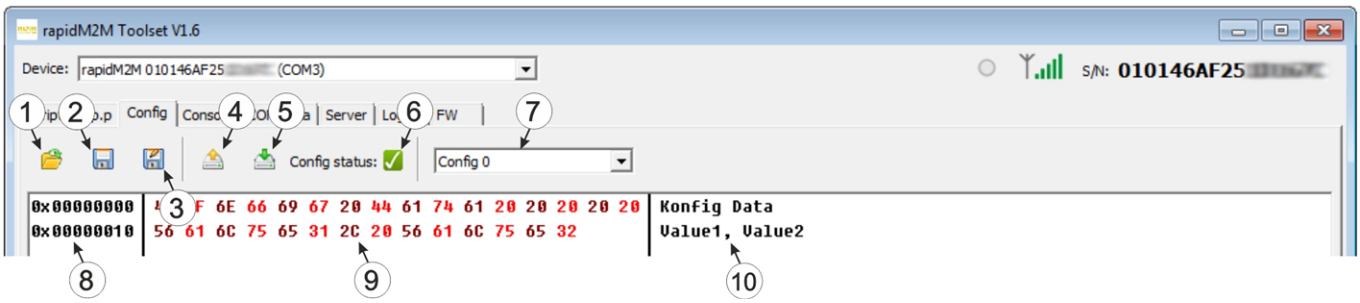
6	Compiles the opened script. It is also automatically saved during this process.
7	Transfers the opened script into the module. During this process, the script is initially saved and then compiled. The compiled PAWN binary (*.amx) is saved in the same folder as the script.
8	Status of the script transfer(see "Script state")
9	Size of the compiled code
10	Specifies the assignment of the available memory as a %
11	Specifies the size of the available memory on the M120 module (specification of the compressed size)
12	Script editor
13	Debug output of the script compiler. If you double click on an entry in the debug output, the cursor will jump to the relevant line in the script editor.

Script status

Symbol	Description
	Script was successfully loaded in the M120 module.
	Script is currently being transferred to the M120 module.
	The time stamp of the compiled script saved in the M120 module does not match the compiled script of the rapidM2M Toolset .

9.7 "Config" tab

This tab provides access to the 10 configuration memory blocks. These can be edited, saved and transferred from and to the M120 module. The tab is also available if there is no USB connection to a M120 module. However, in this case it is not possible to transfer the configuration blocks to and from the M120 module.



"Config" tab

1	Opens a saved configuration memory block	6	Status of the configuration block transmission (see "Config status")
2	Saves the current open configuration memory block	7	Selection of the configuration memory block[1)]
3	Opens the dialogue to save the current configuration memory block that is open under a different file name	8	Address column
4	Loads the selected configuration memory block from the M120 module	9	Hex display of the configuration memory block
5	Transfers the selected configuration memory block to the M120 module	10	ASCII display of the configuration memory block

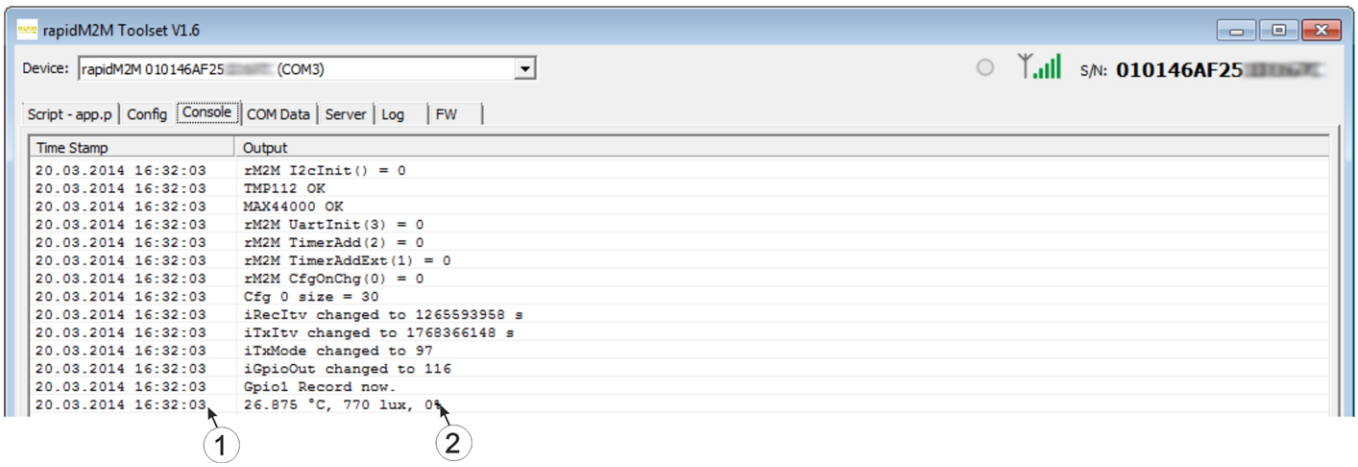
[1)] The "Open", "Save" and "Transfer" operations always only refer to the selected configuration memory block.

Config status

Symbol	Description
✅	The selected configuration memory block was successfully loaded in the M120 module.
⚙	The selected configuration memory block is just being transferred.
❗	The time stamp of the configuration memory block saved in the M120 module does not match that of the rapidM2M Toolset . The display applies to the currently selected configuration memory block.

9.8 "Console" tab

This area is designed to display the debug information that was printed to the standard output within the PAWN scripts by means of the "print" and "printf" functions (see "Console functions" on page 197). The "Console" tab is only available if there is a USB connection to a M120 module.



"Console" tab

1 Time when the debug information was received via the USB interface	2 Display of the debug information

A click with the right mouse button on one of the entries in the debug display will bring up a context menu enabling you to delete or save the list.

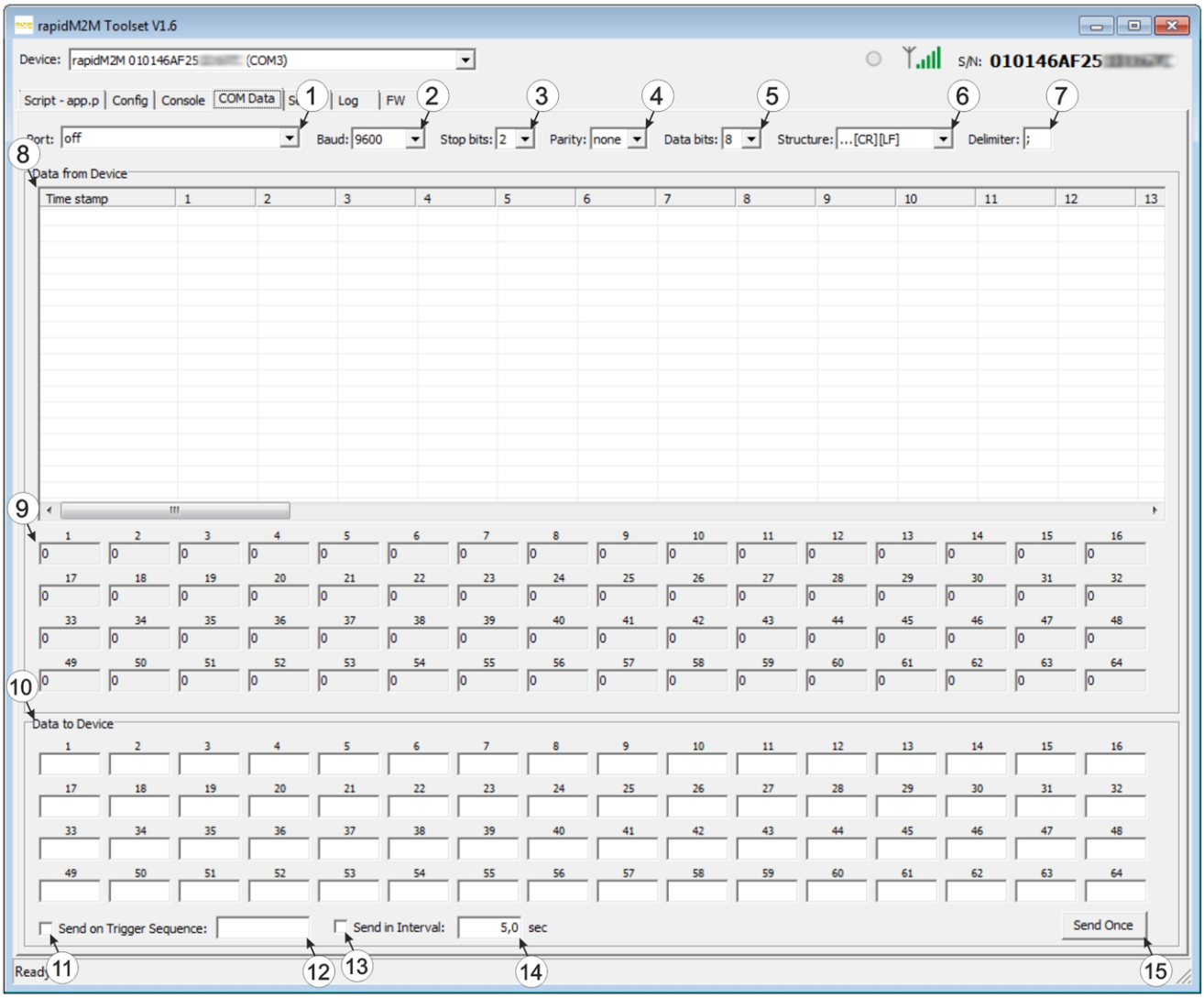
9.9 "COM Data" tab

This tab is designed to simulate a simple serial sensor. An ASCII data message that is split into individual fields by an adjustable delimiter is generated during this process. The frame structure (start and end marker) can also be selected. There are three ways to issue the configured frame via the selected COM interface of the PC:

- Once via the *"Send Once"* button
- At a set interval (*Send in Interval*)
- Upon receipt of a certain sequence (*Send on Trigger Sequence*)

If this option is going to be used, the rapidM2M developer package M120 must be connected to the COM interface of the PC by means of the null modem cable included in the scope of supply.

The "COM Data" tab is also available if there is no USB connection to a M120 module.



"COM Data" tab

1	Selection of the COM interface via which the frame should be sent	9	Display of the values from the last received frame
2	Baud rate to be used	10	Template for inputting the values to be sent
3	Number of stop bits to be used	11	The frame is sent by the PC following receipt of the trigger sequence.
4	Selection of the parity to be used	12	Sequence upon receipt of which the frame must be sent
5	Number of data bits to be used	13	The frame is sent cyclically
6	Configuration of the frame structure (start and end marker), see "Frame Structure"	14	Interval during which the frame should be sent cyclically (minimum value 0.01 sec.)
7	Delimiter to be used	15	Initiates the single transmission of the frame
8	Display of the received frame[1)]		

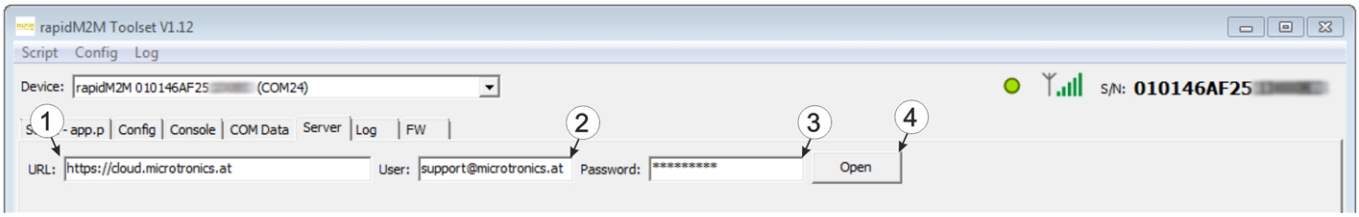
[1)] A click with the right mouse button on one of the entries in the display of received frames will bring up a context menu enabling you to delete or save the list.

Frame structure

Structure	Description
...[CR][LF]	The frame starts with a break and ends with the character string [CR][LF] (oxoD)(oxoA).
...[ETX]	The frame starts with a break and ends with [ETX] (ox03).
[STX]...[CR][LF]	The frame starts with [STX] (ox02) and ends with the character string [CR][LF] (oxoD)(oxoA).
[STX]...[ETX]	The frame starts with [STX] (ox02) and ends with [ETX] (ox03).

9.10 "Server" tab

The REST-API test page of the Cloud server is accessed via this tab. This tab can be used to quickly and simply check whether the complete process chain, starting with saving the data via the script, through to the transfer per GPRS and storage of the data on the server, works correctly. This function can also be used without the M120 module connected via the USB interface. In this case, the serial number of the device must be entered in addition to the server address, user name and password (see ""Server" tab without connected M120 module" on page 149).



"Server" tab with connected M120 module

1	Address of the server (Cloud server)	3	Password for the user account
2	User name of an user account created on the Cloud server	4	Opens the REST-API test page of the Cloud server in the browser

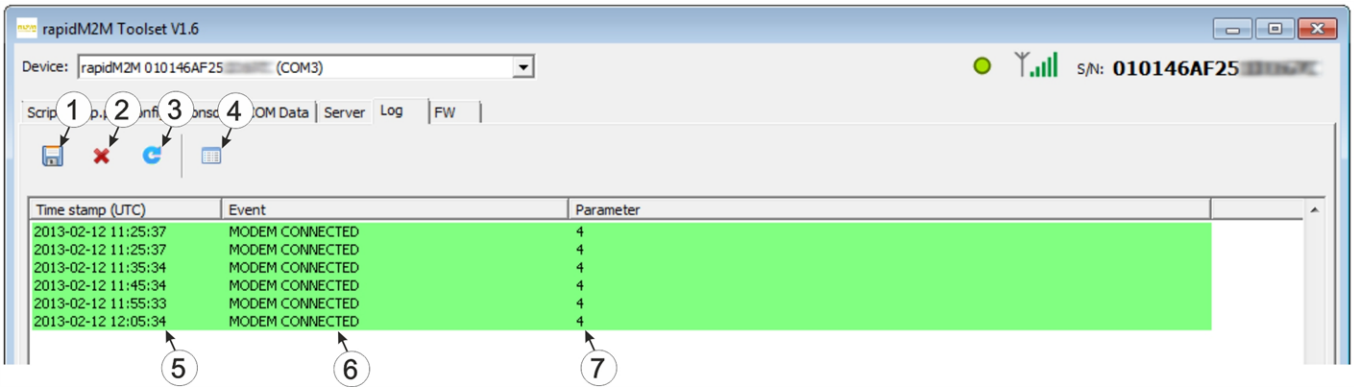


"Server" tab without connected M120 module

1	Serial number of the M120 module for which the data should be accessed via REST-API

9.11 "Log" tab

This area is designed to manage log entries. It enables the entries to be loaded from the M120 module, to be saved as a *.tsv file and entries to be deleted from the memory of the M120 module.

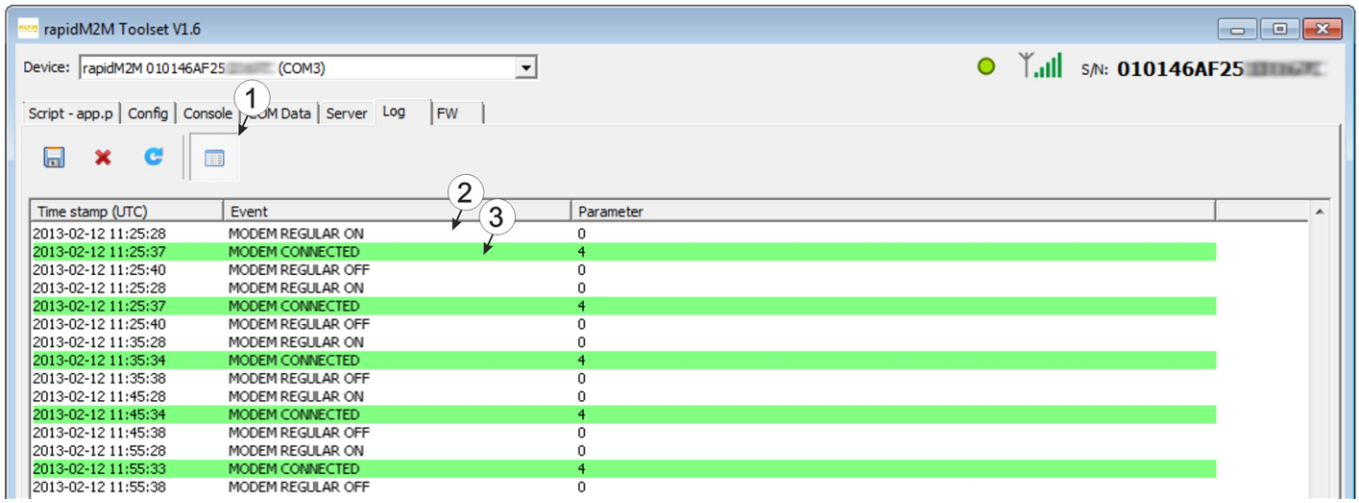


"Log" tab

1	Saves the displayed log entries as a *.tsv file	5	Time stamp of the log entry
2	Deletes the log entries from the memory of the M120 module	6	Log entry
3	Loads the log entries from the M120 module	7	Parameter of the log entry
4	Activates the detailed display of the log entries		

The coloured highlighting indicates how critically the log entry must be evaluated. 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 151).

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

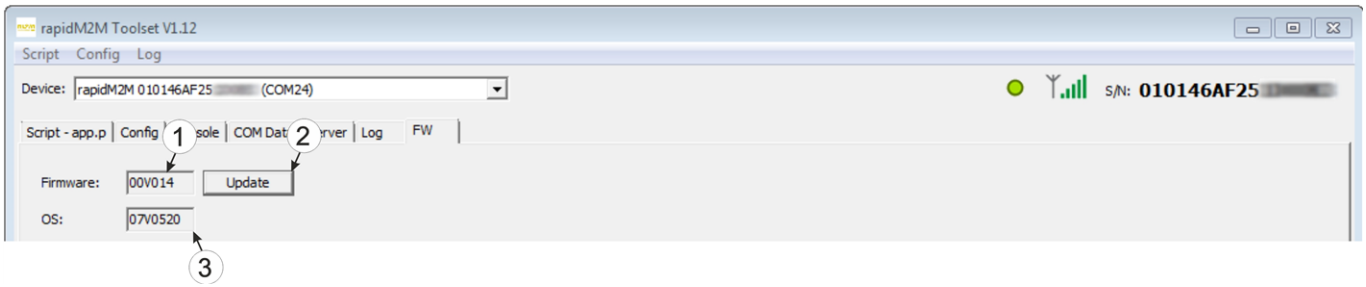


"Log" tab with detailed view activated

1	Detailed display of the log entries activated	3	Log entry that is always displayed
2	Informative log entry that is only visible when the detailed display has been activated		

9.12 "FW" tab

This area enables firmware to be installed directly via the USB interface.



"FW" tab

1	Currently installed firmware	3	OS version of the modem
2	Opens the dialogue to select the firmware file		

9.13 Hot keys

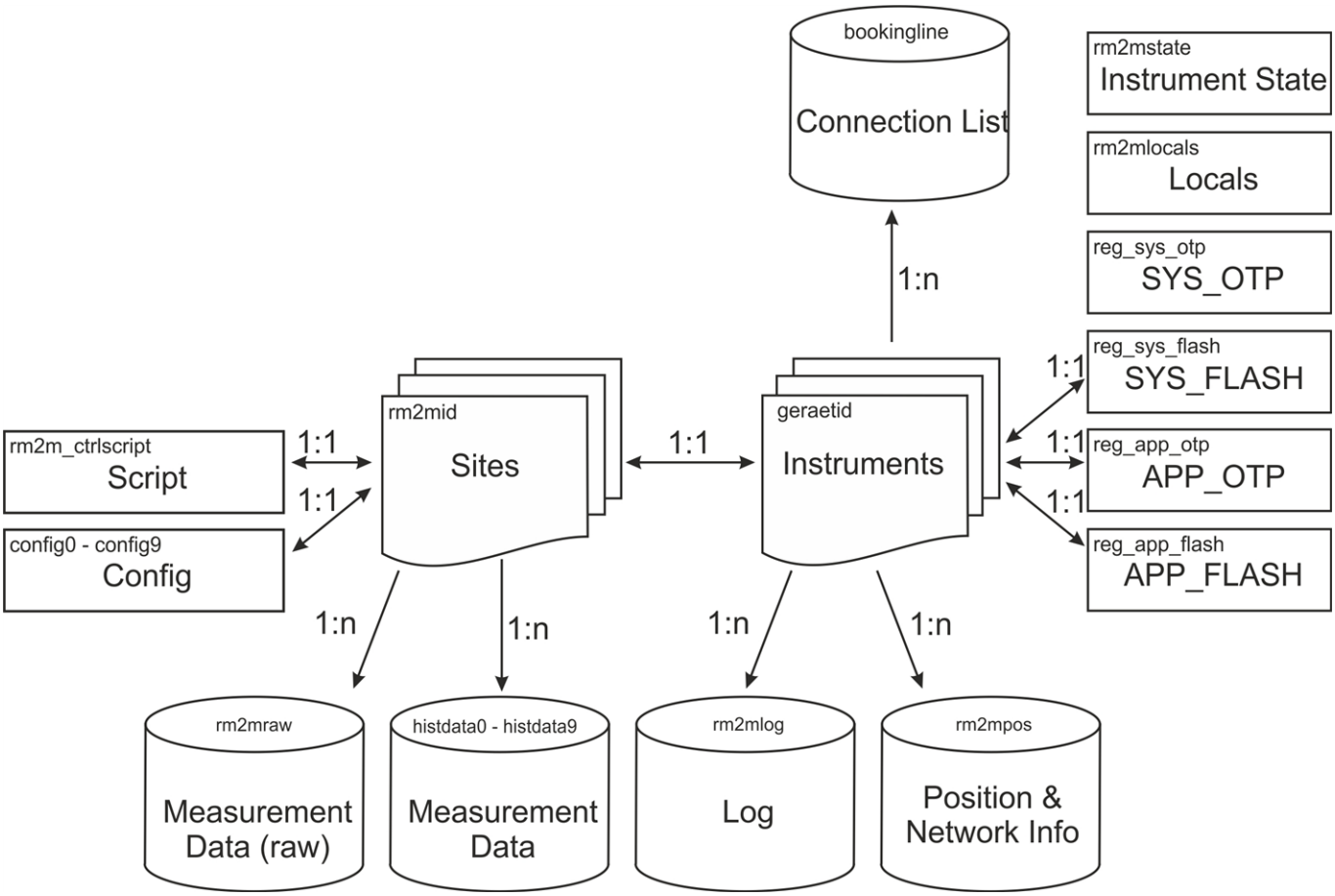
Command	Shortcut	Explanation
New script	Ctrl+N	Creates a new empty script
Open script	Ctrl+O	Opens a saved script
Save script	Ctrl+S	Saves the script that is currently open
Compile script	Shift+F9	Compiles the opened script. It is also automatically saved during this process.
Compile and send to device	F9	Transfers the opened script into the module. During this process, the script is initially saved and then compiled. The compiled PAWN binary (*.amx) is saved in the same folder as the script.

Chapter 10 XML interface

10.1 General

The XML interface is provided to export data from and import data to the Cloud server. However, this is not just limited to the pure measurement data but includes all of the data provided by the Cloud server (e.g. measurement data, configurations, log entries, alarm lists, etc.). It is therefore possible for the customer to completely dispense with the interface of the Cloud server and to create his own user interface. A specially developed PC program or web interface can, for example, be used for this purpose.

10.2 Structure of the database



Structure of the database

Generally applicable table identifiers

Identifier	Description
rm2mid	Site configuration (see "rm2mid" on page 156)
geraetid	Devices (see "geraetid" on page 159)
bookingline	Connection list (see "bookingline (read only)" on page 161)

Device-specific table identifiers

Identifier	Allocation	Description
histdata0	Site	Measurement data 0 (structured) (see "histdata0 - histdata9" on page 162
histdata1	Site	Measurement data 1 (structured)
histdata2	Site	Measurement data 2 (structured)
histdata3	Site	Measurement data 3 (structured)
histdata4	Site	Measurement data 4 (structured)
histdata5	Site	Measurement data 5 (structured)
histdata6	Site	Measurement data 6 (structured)
histdata7	Site	Measurement data 7 (structured)
histdata8	Site	Measurement data 8 (structured)
rm2mlog	Device	Device log (see "rm2mlog" on page 162)
rm2mstate	Device	Device status (see "rm2mstate" on page 162)
rm2mlocals	Device	Localisation information (see "rm2mlocals" on page 163)
rm2mpos	Device	Position and network information (see "rm2mpos" on page 163)
rm2m_ctrlscript	Site	Byte code of the script (see "rm2m_ctrlscript" on page 164)

Identifier	Allocation	Description
reg_sys_otp	Device	System information that is written once as part of the production process (see "reg_sys_otp" on page 165)
reg_sys_flash	Device	System information that must be able to be changed during operation (see "reg_sys_flash" on page 165)
reg_app_otp	Device	Application-specific information that is written once as part of the production process (see "reg_app_otp" on page 165)
reg_app_flash	Device	Application-specific information that must be able to be changed during operation (see "reg_app_flash" on page 166)
config0	Site	Configuration memory block 0 (see "config0 - config9" on page 166)
config1	Site	Configuration memory block 1
config2	Site	Configuration memory block 2
config3	Site	Configuration memory block 3
config4	Site	Configuration memory block 4
config5	Site	Configuration memory block 5
config6	Site	Configuration memory block 6
config7	Site	Configuration memory block 7
config8	Site	Configuration memory block 8
config9	Site	Configuration memory block 9
rm2mraw	Site	Measurement data (raw format) (see "rm2mraw" on page 166)

The device-specific table identifiers are distinguished as ones that are assigned to the device (geraetid) and ones that are assigned to the site (rm2mid). If the assignment of the device and site is deallocated, the tables that are assigned to the device (e.g. the device log) remain firmly linked to the device and the tables that are assigned to the site (e.g. measurement data) remain firmly linked to the site. This means that if a new device is assigned to a site, the measurement data that was generated while the previously assigned device was connected to the site can be read out (if the "geraetid" table of the new device is selected as the starting point).

The structure of the database must be known to create a xPath query (see "Syntax of the xPath query" on page 167). If, for example, the measurement data for a certain serial number should be

read out, the path initially goes via the "geraetid" table that contains the serial number, to the "rm2mid" table that contains the link between the device and site and to the "rm2mraw" table that contains the required measurement data. An example is provided in t.b.d.

10.2.1 Generally applicable table identifiers

10.2.1.1 rm2mid

This table includes all of the site-specific configurations..

name

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

timezone

Regional settings (not relevant for raw measurement data as this is stored in the UTC)

Value	*Explanation*
99.00	*The regional setting for the site is adopted from the global server setting.*
-12.00	*UTC -12 hours*
:	*:*
+14.00	*UTC +14 hours (Kiritimati)*

summertime

Configuration for automatic time adjustment

Value	*Explanation*
DEFAULT	*The configuration for the time adjustment is adopted by the global server setting*
OFF	*Automatic time adjustment deactivated*
USA;99CC96152B000000	*Predefined setting for the American area*
EU;99CC95A627080884	*Predefined setting for the European area*

alert_mode

Setting for acknowledging the alarms

Value	*Explanation*
0	*The global server setting is used to determine whether alarms must be acknowledged automatically or manually.*
1	*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.*
2	*Alarms must be acknowledged by the user.*

max_transfer_volume

Settings for the transfer volume alarm

Value	*Explanation*
-1	*The setting for the transfer volume alarm is taken from the global server settings.*
0	*The transfer volume alarm is deactivated.*
› 0	*Threshold at which the transfer volume alarm should be triggered [KB]*

max_transfer_volume_kb

This is a "Ghost" field, the content of which always corresponds to that of the "max_transfer_volume" field.

comment

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

loc_site_lat

Specification of the geographical latitude where the site is located [°]

loc_site_long

Specification of the geographical longitude where the site is located [°]

loc_alarm_rad_mode

A radius can be defined for a site. Depending on the settings for this parameter, an alarm is triggered if the device leaves this radius.

Value	*Explanation*
0	*Radius monitoring deactivated (no alarm)*
› 0	*Specification of the alarm radius [km]. An alarm is triggered if the device leaves the radius.*

loc_alarm_rad

This is a "Ghost" field, the content of which always corresponds to that of the "loc_alarm_rad_mode" field.

view

Selection of the report that is loaded by clicking on the device link in the maps

Value	*Explanation*
OFF	*The default graphic is loaded.*
"viewid"	*The selected report is loaded.To obtain the "viewid", open the required report (see ""Sites" area" on page 118) and take the ID from the address line of the browser. (e.g.. . . . `&viewid=`**`A1A5AD6CD299EF96`**`&`. . . .).*

ftpexportprofile

Selection of the FTP export profile

Value	*Explanation*
0	*FTP export deactivated*
"id"	*To obtain the "id", open the configuration of the required FTP export profile (see myDatanet Server Manual 805002) and take the ID from the address line of the browser. (e.g. . . . .&id=**A0B66D1CF7ED87DD**& . . . .).*

ftpexportdir

makes overwriting the standard directory of the selected FTP export profile possible [0-100 characters]

mappincaption

Specifies whether the title of the site should be displayed in the "Map"

mapovpincaption

Specifies whether the title of the site should be displayed in the "Overview map"

dbdesc

Configuration of the myDatanet Connector (see "myDatanet Connector " on page 277) (max. 40kB)

nameassitetype

Specifies whether the name of the application template from which the site was created should be displayed as the site type in the site overview

Value	*Explanation*
0	*Use type of the site*
1	*Use name of the application template*

app_version

Current version of the application template. The number is automatically increased each time new settings are saved. (read only)

app_min_fw_modem

The minimum required modem version can be specified for the application template.

Four hex-coded characters. The first two characters specify the main version number and the last two characters specify the subversion number (e.g. 010B for 01v011).

app_min_fw_ctrl

The minimum required controller version can be specified for the application template.

Four hex-coded characters. The first two characters specify the main version number and the last two characters specify the subversion number (e.g. 010B for 01v011).

app_public
Specifies whether the application template is available for all customers.

Value	*Explanation*
0	*Availability limited to individual customers*
1	*Available to everyone (default)*

mapview_showvaluesinmap
Specifies whether the measurement values and the title of the site should be displayed in the map view.

Value	*Explanation*
0	*Off (default)*
1	*Display measurement values and site title*

mapview_clicktarget
Reserved for extensions

mapview_clicktarget_view
Reserved for extensions

cho - ch4
Reserved for extensions

binary_src
Specifies whether the last script was entered directly via the server interface or was uploaded as a file

Value	*Explanation*
0	*The script was entered via the server interface.*
1	*The script was uploaded as a binary file (*.amx).*

kunde
Unique data record number of the customer to whom the site is assigned (read only)

device
Unique data record number of the device to which the site is assigned (read only)

template
Unique data record number of the application template from which the site was created (read only)

10.2.1.2 geraetid

This table includes general information and configurations that are the same for all devices of the myDatanet range.

kunde
Unique data record number of the customer to whom the device is assigned (read only)

conn_path

Specifies the communication channel via which the last connection was established

Value	*Explanation*
0	*GSM/GPRS*
1	*Other communication channel (e.g. LAN)*

devsn

Serial number of the instrument

devclass

Device class The instrument class of the site and instrument must match for an instrument to be able to be connected to a site.

telno

Telephone number of the SIM card. The control SMS messages (e.g. wakeup) are sent to this number. Format: +43555837465[0-20 characters]

devflags

Additional information regarding the instrument class (for internal use) (read only)

swversion

Current software version installed on the measurement controller (read only)

modemversion

Current software version installed on the modem controller(read only)

osversion

OS version of the modem(read only)

lastcon

Time stamp of the last connection (read only)

lastwakeup

Time stamp of the last wakeup (read only)

lastdisc

Time stamp of the last disconnection (read only)

lasterr

Time stamp of the last transmission error (read only)

lastaloha

Time stamp of the last Aloha connection (read only)

fw_update

Settings for the firmware update

Value	*Explanation*
0	*Off: Firmware update is deactivated.*
1	*On: As soon as a new version of the selected firmware type is available, this is installed immediately.*
2	*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!).*
3	*Allow downgrade: Facilitates the installation of an older firmware version than the one on the device (NOT RECOMMENDED!)*
4	*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".*
5	*Ignore: The firmware update is deactivated and no information is provided about available firmware updates.*

fw_release_type

Settings for the firmware type

Value	*Explanation*
0	*Released: Only firmware versions that have successfully undergone internal and field testing are installed (this practically eliminates malfunctions).*
1	*Release candidate: Only firmware versions that have successfully undergone internal testing are installed (malfunctions cannot be excluded).*
2	*Beta release: Even firmware versions that have not successfully undergone all of the internal tests are installed (malfunctions may occur).*

10.2.1.3 bookingline (read only)

This table includes information about the connections between the devices and the Cloud server

kunde

Unique data record number of the customer to whom the device is assigned

geraet

Unique data record number of the device to which the connection list is assigned

text

Duration of the connection

context

Title of the site with which the device is linked or error that occurred during the connection

source

Serial number and IP address

cmh_quantity
Amount of received data in bytes (device to the server)

cmh_code
Indicates whether the device has established a connection to the Cloud server (41=closed, 942=connected)

> ***Important note:*** *Table entries with different codes to the ones mentioned above include information that is only intended for internal use and that should be filtered out by the customer.*

10.2.2 Device-specific table identifier

10.2.2.1 histdatao - histdata9

These tables contain the measurement data in a structured format. The Cloud server is informed about the structure of the measurement data blocks of the "rm2mraw" table using the myDatanet Connector (seeCloud"myDatanet Connector " on page 277) and the measurement values are then copied into the "histdatao" table, for examplemyDatanet Connector "myDatanet Connector " on page 277. This is why the data can be used for reports, views, graphics, etc. on the Cloud server.

flags
Reserved for extensions

cho - ch999
Measurement values that were defined using the myDatanet Connector (see "myDatanet Connector " on page 277) and were copied from the "rm2mraw" table

10.2.2.2 rm2mlog

This table contains the log entries and error codes (see "Log entries and error codes " on page 289) of the device.

log
Log entry

par
Parameter for the log entry

10.2.2.3 rm2mstate

This table contains the device status at the time of the last transmission. All of the fields in this table can only be read and not changed (read only).

gsm_level
GSM level [dBm]

com_mode
Connection type at the time of the last connection

sim_id
IMSI of the SIM chip

sim_apn
APN that was used for the last connection

net_mcc
MCC (mobile country code) of the network that was used for the last connection

net_mnc
MNC (mobile network code) of the network that was used for the last connection

imei
IMEI of the modem

sim_icc_id
ICC ID of the SIM chip

idstring
rapidM2M module identification (e.g. "rapidM2M M1 HW1.1")

10.2.2.4 rm2mlocals

This table contains the localisation information.

time_offset
Difference (in seconds) between the system time (UTC) and the local time configured for the site on the Cloud server (incl. consideration of summer/winter time).

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

10.2.2.5 rm2mpos

This table contains the position and network information. The generation of this data can be initiated via the "rM2M_TxStart() ", "rM2M_SetPos()" or "rM2M_SetPosNMEA()" script function.

gps_lat
Determined geographical latitude where the device is located [°]

gps_long
Determined geographical longitude where the device is located [°]

gps_elev
Determined height above sea level [m]

gps_quality
Specifies the quality of the determined position data. The position for devices without a GPS receiver is calculated based on the information from the GSM cells in the receiving range.

Value	*Explanation*
0, 1	*Position could not be determined.*
10	*Only the country in which the device is located could be determined.*
20, 30	*The position has been limited to a larger area within a country.*
›=100	*Position could be calculated from the information of the GSM cells in the receiving range*
›31,000	*Position determined by the GPS receiver*[1)]

[1)] *The device does not include a GPS receiver.*

state
Reserved for internal use

Information about the five ten GSM cells in the receiving range that were used to determine the position

gsm_mcc_0 - gsm_mcc_4 9
MCC (mobile country code) of the GSM cell

gsm_rssi_0 - gsm_rssi_4 9
Received GSM level [dBm] for the GSM cell

gsm_mnc_0 - gsm_mnc_4 9
MNC (mobile network code) of the GSM cell

gsm_ta_0 - gsm_ta_4 9
TA (timing advance) of the GSM cell

gsm_lac_0 - gsm_lac_4 9
LAC (location area code) of the GSM cell

gsm_cellid_0 - gsm_cellid_4 9
Cell ID of the GSM cell

10.2.2.6 rm2m_ctrlscript

This table contains the script type selection (see "Control" on page 189) as well as the script binary loaded on to the server in hex notation (2 characters per byte), that are transferred to the rapidM2M module M120 .

script_error

Indicates a faulty script (must not be changed via XML)

Value	*Explanation*
0	*The script is executed correctly.*
1	*The script was deactivated by the device due to a runtime error. More detailed information is provided in the device log (see "Pawn script error codes" on page 251).*

script_type

Selection of the script type (must not be changed via XML)

Value	*Explanation*
0	*Script deactivated*
2	*Activates script processing and informs the rapidM2M module M120 that the script is a "Pawn script" (see "Pawn script" on page 193).*

10.2.2.7 reg_sys_otp

This table contains the System information that is written once as part of the production process (read only).

json

Content of the "REG_SYS_OTP" registration memory block as a string using the JSON notation (see "Registration memory blocks" on page 62)

10.2.2.8 reg_sys_flash

This table contains the System information that must be able to be changed during operation (read only).

json

Content of the "REG_SYS_FLASH" registration memory block as a string using the JSON notation (see "Registration memory blocks" on page 62)

10.2.2.9 reg_app_otp

This table contains the application-specific information that is written once as part of the production process (read only).

json

Content of the "REG_APP_OTP" registration memory block as a string using the JSON notation (see "Registration memory blocks" on page 62)

10.2.2.10 reg_app_flash

This table contains the application-specific information that must be able to be changed during operation (read only).

json
Content of the "REG_APP_FLASH" registration memory block as a string using the JSON notation (see "Registration memory blocks" on page 62)

10.2.2.11 configo - config9

This table contains the customer's freely definable, independent memory blocks each with 10kB for the configuration data. The Cloud server must be aware of the logical structure of the configuration data blocks for the configuration parameters to be able to be edited or displayed via the interface of the Cloud server. This is completed by means of the myDatanet Connector (see "myDatanet Connector " on page 277).

Fieldo - Field999
Configuration parameter defined via the myDatanet Connector (see "myDatanet Connector " on page 277).

10.2.2.12 rm2mraw

This table contains the measurement data in raw format. The Cloud server is not aware of the logical structure of the measurement data block. To use the data for reports, views, graphics, etc. on the Cloud server they must, for example, be copied to the "histdatao" table using the myDatanet Connector (see "myDatanet Connector " on page 277).

flags
Reserved for extensions

data
Measurement data block (raw format)

10.3 XML export

10.3.1 Configuration of a HTTP-GET command

A request is executed by means of a HTTP-GET command that corresponds to simply entering the command in the address line of a standard Internet browser. The command line (URL) is assembled as follows and provides an XML document as a result:

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

Parameter	Explanation
{URL}	Address of the server on which the xPath module operates (Cloud server)
{Port}	Port on which the xPath server operates. The standard setting is 8080.
{xPath Query}	xPath of the required XML table (e.g. "Structure of the database" on page 153). A description of the syntax is provided in "Syntax of the xPath query" on page 167.
{login}	Username and password of a user account created on the Cloud server to authenticate the request (e.g. `user=u1&password=p1`).
{settings}	OPTIONAL Settings relating to the output format of the request. There is only one possible parameter. It deactivates the display of the field attributes: `attributes=off` Settings relating to the output of the data. There is only one possible parameter. It specifies the data compression period: `period=...` (see "Data record compression" on page 172)
{update}	OPTIONAL Update of data fields in the query result (e.g. `&$field01=33`) (see "Updating data fields" on page 174)
{data_type}	OPTIONAL Selection of the output format: **0** *XML document (see "Format of the XML result" on page 177)* **1** *JSON document (see "Format of the JSON result" on page 178)* The XML document is used as the output format if this parameter is not specified.
{ verbose_time }	OPTIONAL **1** *The time stamps are issued with an increased accuracy (ms) (e.g. `stamp =" 23.5.2012 15:21:09.348 "`).*

10.3.1.1 Syntax of the xPath query

10.3.1.1.1 List of the fields in a table

A list of the fields within a table is read out via an `@` before the table identifier.

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

Example above provides a list of the fields in the "Device" table (*geraetid*):

```
<Result>
  <Field title="Customer">customer</Field>
  <Field title="Serial number">devsn</Field>
  <Field title="Device class">devclass</Field>
  <Field title="Firmware version">swversion</Field>
  <Field title="Modem version">modemversion</Field>
  <Field title="OS version">osversion</Field>
  <Field title="Last connection" units="<%data locals.time_code%> 
    <%data locals.offset_record.lastcon%>">lastcon</Field>
  <Field title="Last wakeup" units="<%data locals.time_code%> 
    <%data locals.offset_record.lastwakeup%>">lastwakeup</Field>
  <Field title="Last disconnection" units="<%data locals.time_code%> 
    <%data locals.offset_record.lastdisc%>">lastdisc</Field>
  <Field title="Last transmission error" units="<%data locals.time_code%> 
    <%data locals.offset_record.lasterr%>">lasterr</Field>
  <Field title="Last Aloha connection" units="<%data locals.time_code%> 
    <%data locals.offset_record.lastaloha%>">lastaloha</Field>
  <Field title="Firmware update">fw_update</Field>
  <Field title="Firmware type">fw_release_type</Field>
</Result>
```

A list of possible table identifiers is provided in "Structure of the database" on page 153.

10.3.1.1.2 List of site types

You can access a list of all the types of sites that are supported by the Cloud server by specifying the xPath query *SiteTypes*:

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

Result:

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

10.3.1.1.3 Filter conditions

The filter condition is always within square brackets `[]` and comprises expressions that can be linked by keywords such as *and* and *or*. The expressions can be prioritised by using round brackets `()` . Only *and* links are permissible at the highest level. A filter expression comprises a field name, an operator and a constant. All of the fields in the table can be used.

> ***Note:*** *Upper and lower case are taken into consideration for the name of the site. The condition must not be specified with an inverted comma ' when a filter is applied to* `id` *or* `ref`*.*

The following attributes are available in addition to the fields from the table:

Attribute	Explanation
id	Unique data record number
stamp	Time stamp of the data record (only operator ›= and ‹= are permitted)
ref	Unique data record number of the reference data record

The following operators are supported: **=, ‹›, ›, ‹, ›=, ‹=**

The following types can be used as constants:

Data type	Explanation
Integer	Whole numbers
Float	Floating point numbers
Strings	Text indicated by single inverted commas '
Date	Time stamp within single inverted commas '

Formats (comma, date format, etc.) are adopted from the user or server settings.

10.3.1.1.3.1 Example:

The configuration of all sites of the type rapidM2M is displayed if no filter conditions are provided.

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

Result:

```
<Result>
  <rm2mid id="A1526911CF9CCAA0" stamp="8.2.2013 11:48:16">
    <tag>FFFF6700</tag>
    :
    <name title="Name"> site1 </name>
    :
  </rm2mid>
  <rm2mid  id="A1617F03900A44F3"  stamp="8.2.2013 11:54:52" >
    <tag>FFFF6700</tag>
    :
    <name title="Name"> site2 </name>
    :
  </rm2mid>
</Result >
```

The filter must be set up as follows if only the configuration for the "Site1" site has to be returned:

```
https://cloud.microtronics.at:8080/rm2mid[name='Site1']?user=u1&password=p1
```

Result:

```
<Result>
  <rm2mid id="A1526911CF9CCAA0" stamp="8.2.2013 11:48:16">
    <tag>FFFF6700</tag>
    :
    <name title="Name"> site1 </name>
    :
  </rm2mid>
</Result >
```

10.3.1.1.4 Fields

If more than one field is read out, the list of fields must be placed in round brackets `()` and separated by a `|` or `blank space`. If none of the fields are specified, all of the available fields are issued if this is permitted by the user's authorisation level.

> ***Note:*** *The special `tag` field can be requested in every table and it returns the tag from the requested table (e.g. FFFF0005 for the `geraetid` table identifier).*

10.3.1.1.4.1 Example

The following example provides the names of all sites of the type rapidM2M

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

Result:

```
<Result>
  <rm2mid id="A1526911CF9CCAA0" stamp="8.2.2013 11:48:16">
    <name title="Name">Site1</name>
  </rm2mid>
  <rm2mid id="A1617F03900A44F3" stamp="8.2.2013 11:54:52">
    <name title="Name">Site2</name>
  </rm2mid>
</Result>
```

10.3.1.1.5 Data retrieval including index specifications

Only the following index specifications are supported:

Index	Explanation
[0]	First data record (with the oldest time stamp)
[last()-1]	Last data record (with the latest time stamp)

Other index specifications, such as `[5]` or `[last()-2]` are invalid.

10.3.1.1.5.1 Example

The following example provides the first data record of a rapidM2M type site that is called "Site1".

```
https://cloud.microtronics.at:8080/rm2mid[name='Site1']/rm2mraw[0]
?user=u1&password=p1
```

Result:

```
<Result>
  <rm2mraw id="F000002D00000000" stamp="11.2.2013 14:39:31">
    <tag>000067FF</tag>
    <flags title="Flags">0000000000000000</flags>
    <data title="data">0000000018AD029241D88000443D000000000
       001000000000000000000000</data>
  </rm2mraw>
</Result>
```

Explanation:

rm2mid
Specifies that the data of a rapidM2M type site should be queried

[name='Site1']
Specifies that the name of the site, that has to be queried, is "Site1".

rm2mraw
Specifies that the data from the measurement data table should be queried.

[0]
Specifies that the first data record (with the oldest time stamp) should be queried.

The following example supplies the last data record of a rapidM2M type site that is connected to the device with serial number "010146AF25XXXXXX".

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

Result:

```
<Result>
  <rm2mraw id="F000002D00000018" stamp="11.2.2013 15:02:42">
    <tag>000067FF</tag>
    <flags title="Flags">0000000000000000</flags>
    <data title="data">0000000318AD080141D98000443540000000000
      000000000000000000000000</data></rm2mraw>
</Result>
```

Explanation:

geraetid
Specifies that the entry point for the request is the device table

[devsn='010146AF25XXXXXX']
Specifies that the serial number for the device, for which the date has to be queried, is "010146AF25XXXXXX".

rm2mid
Specifies that the data of a rapidM2M type site should be queried.

rm2mraw
Specifies that the data from the "Measurement data" table should be queried.

[last()-1]

Specifies that the last data record (with the latest time stamp) should be queried.

10.3.1.2 Data record compression

Note: *JSON documents cannot be released when data record compression is activated.*

Data record compression can be activated via the optional `period=...` setting. This means that all of the data records within the specified period are compressed with the per field adjustable calculation method.

The calculation method must be specified in brackets after each field in the field list (see "Fields" on page 170) of the xPath query:

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

The example above provides three different results for the values of channel 0 compressed over 15 min.:

- cho(method=1): Minimum within the compression period
- cho(method=2): Maximum within the compression period
- cho(method=16;sum_mode=4): Daily total over the compression period

Assumption: The value of channel 0 increases by one per minute.

```
<Result>
  <record stamp="2013-01-25 11:30:00">
    <ch0_min>0.0</ch0_min>
    <ch0_max>14.0</ch0_max>
    <ch0_sum>105.0</ch0_sum>
  </record>
  <record stamp="2013-01-25 11:45:00">
    <ch0_min>15.0</ch0_min>
    <ch0_max>29.0</ch0_max>
    <ch0_sum>330.0</ch0_sum>
  </record><record stamp="2013-01-25 12:00:00">
    <ch0_min>30.0</ch0_min>
    <ch0_max>44.0</ch0_max>
    <ch0_sum>555.0</ch0_sum>
  </record>
</Result>
```

`period=...`

Period in minutes to which it should be compressed:

Example:

60	*hourly data*
1440	*daily data*
10080	*weekly data*
44640	*monthly data*
133920	*quarterly data*

`method=…`

The compression methods can be specified per channel:

1	*Minimum within the compression period*
2	*Maximum within the compression period*
4	*Average over the compression period*
8	*Median over the compression period*
16	*Total over the compression period*

Several methods can be used simultaneously. Individual values must be added up for this purpose. That means that `(ch0(method=1) | ch0(method=2))` *is equivalent to* `(ch0(method=3))`.

If the compression method "Total" (16) is activated, the summation method `sum_mode=…` *must be specified in addition to the* `method=…` *compression method.*

`sum_mode=…`

1	*1/sec (e.g. one channel is recorded in m^3/s; the total period equals m^3.)*
2	*1/min (e.g. one channel is recorded in m^3/min; the total period equals m^3.)*
3	*1/h (e.g. one channel is recorded in m^3/h; the total period equals m^3.)*
5	*1/day (e.g. one channel is recorded in m^3/day; the total period equals m^3.)*
4	*Total of a day counter (day counter means that the counting channel is reset once per day). The difference between the values ($_n$ value - $_{n-1}$ value) is added up over the compression period. The reset time of the day counter is taken into consideration during this process.*
6	*Total of an interval counter (interval counter means that the counting channel is reset after every measurement). The values are added up over the compression period.*

10.3.1.3 Updating data fields

A HTTP-GET command can also be used to update the fields of a data record. For this purpose, the HTTP-GET command must be extended by the relevant field assignment starting with a dollar sign `$` . Any number of fields can be overwritten at once.

> ***Note:*** *The user must have the relevant rights to modify the data (see tbd).*

10.3.1.3.1 Example

The following example sets the "Firmware update" to "ON" and the "Firmware type" to "Release candidate" for all of the devices that are compatible with the rapidM2M type site.

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

Result:

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

Explanation:

geraetid
Specifies that the entry point for the request is the device table.

[devclass=6700]
Specifies that the action should apply to all of the devices that are compatible with the rapidM2M type site.

$fw_update=1
Specifies that the "Firmware update" should be set to "On".

$fw_release_type=1
Specifies that the "Firmware type" should be set to "Release candidate".

The following example sets the "Firmware Update" to "Off" for the device with serial number "010146AF25XXXXXX".

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

Result:

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

Explanation:

geraetid

Specifies that the entry point for the request is the device table.

[devsn='010146AF25XXXXXX']

Specifies that the action should apply to the device with serial number "010146AF25XXXXXX".

$fw_update=0

Specifies that the "Firmware update" should be set to "Off".

10.3.1.4 Data retrieval via streaming technology

In addition to the active retrieval of individual data records a stream can be opened. This stream sends data until it is actively terminated.

A library that handles the streaming itself and the associated characteristics of the various browsers is provided for the streaming technology.

This library is available to download at the following address:

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

streamer_ = new rM2M . Streamer (settings);

Initialise the stream with the previously created settings

Parameter	*Explanation*
settings	*An object with the following characteristics must be transferred to the "Streamer" method to start the streaming process.* *The authentication is completed via the user and password or the userid. This means that the object must either contain the "User" and "Password" characteristics or the "Userid" characteristics.* **protocol** *Protocol that should be used (http or https)* **address** *Address of the server from which the data should be streamed* **port** *Port via which the server should be accessed* **user** *User name of the user created on the server* **password** *Password of the user created on the server* **userid** *ID of the user created on the server.*

streamer_. start (idSN, idContent);

Start streaming

Parameter	*Explanation*
idSN	*Serial number of the device (as a string) from which the data should be streamed*
idContent	*Selection of the data that should be streamed. Only the predefined strings are permitted at this point:* **histdata0 - histdata9** *Measurement data 0-9 (structured)* **config0 - config9** *Configuration memory block 0-9* **rm2mraw** *Measurement data (raw format)*

streamer_ . stop ();

Terminate streaming

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

The stream object supports events that can be registered with the `addEventListener` method.

Parameter	*Explanation*
data	*Name of the event (see "Events" on page 176)*
function (name, param1)	*If an event occurs, the relevant registered function is called up and the following two parameters are assigned to it:* **name** *The name of the event ("trying", "start", etc.). This ensures that all of the events can be processed in one function if necessary.* **parm1** *Includes the relevant data depending on the event. The content of param1 is included in the event explanations.*

10.3.1.4.1 Events

trying

An attempt is made to start a stream with the technology transferred in param1. The available technologies are saved as constants in rM2M.StreamTec.

start

A stream has been started successfully. Param1 includes the technology; this is the same for "trying".

stopping

The stop process of a stream has been initialised. Param1 includes the technology.

stop

The stream was stopped. Param1 includes the technology.

data

Data has been received. Param1 includes an object with the received data.

10.3.1.5 Output formats of the HTTP-GET command

The output format is selected using the optional parameter `{data_type}` (see "Configuration of a HTTP-GET command" on page 166). The XML document is used as the output format if this parameter is not specified.

10.3.1.5.1 Format of the XML result

The result is an XML document for which the root element is `<Result>`. At least one child element is issued with the title of the table for each data record.

The data records generally have the following attributes:

Attribute	Explanation
id	Unique number of the data record
stamp	Time stamp of the data record

The data fields have the following attributes:

Attribute	Explanation
title	Title of the data field (for measurement data this is dependent on the site settings)
units	Units of the data field for measurement data (dependent on the site settings)
calc	OPTIONAL Is only displayed if it is a channel that has been calculated by the sever

10.3.1.5.1.1 Example:

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

10.3.1.5.2 Format of the JSON result

The end result is a JSON document that consists of one or several objects.

Each of these objects comprises four properties:

Properties	Explanation
name	Table identifier
id	Unique identification of the data record
stamp	Time stamp of the data record
fields	Array of objects that represent the fields of the table

The objects that represent the fields of the table have the following properties:

Properties	Explanation
name	Internal field designation
title	Title of the data field (for measurement data this is dependent on the site settings)
units	Units of the data field (for measurement data this is dependent on the site settings)
value	Value of the data field
calc	OPTIONAL Only displayed if it is a channel that has been calculated by the sever (only for measurement data)

Note: *The "name" and "value" properties are always present, while the other properties are only available if required.*

10.3.1.5.2.1 Example

> ***Note:*** *Additional line breaks have been added to the following example to make it easier to read.*

```
{name: 'geraetid', id: 'A15D6246F7AEBFAA', stamp: '13.2.2013 09:10:05',
   fields: [{name: 'tag', value: 'FFFF0005'},
            {name: 'kunde', title: 'customer', value: 'A1519985809987EC'},
            {name: 'devsn', title: 'serial number', value: '010146AF25XXXXXX'},
            {name: 'devclass', title: 'dev%c3%i4ce class', value: '6700'},
            {name: 'swversion', title: 'Firmware%20Version', value: '0000'},
            {name: 'modemversion', title: 'Modem%20Version', value: '0005'},
            {name: 'osversion', title: 'OS%20version', value: '07.0474'}, ... ]}
{name: 'geraetid', id: 'A1526911CF1A42CC', stamp: '13.2.2013 09:33:54',
    fields: [ ... ]}
```

10.4 XML import

> ***Important note:*** *XML import is not available on all Cloud servers. Data cannot be imported if the direct login parameter that clearly identifies the user on the server is not displayed in the input screen for editing the user data (see "myDatanet Server Manual " 805002).*

The XML import is designed to import the data generated or changed via the user interface, created by the customer, back into the Cloud server.

- Only existing measurement data and settings can be changed or new measurement data can be added. No historical measurement data can be added.
- It must be ensured that the `stamp` of the data records is always created in ascending order. Errors can occur if this is not the case.
- If any settings are changed, it must be ensured that the `stamp` is set to the current date/time. Otherwise the data are not synchronised with the device.
- When creating measurement data, the `id="0"` and `<tag>00006700</tag>` (67 representing the device class) must always be specified.
- `<tag>0</tag>` must always be specified during the deleting process. Historical data (e.g. measurement data) cannot be deleted.

10.4.1 Transferring data to the Cloud server

A HTTP post command can be used to transfer a XML file to be imported by the Cloud server.

A HTTP POST command can, for example, be executed with the freely available "Wget" command line tool (http://www.gnu.org/software/wget/).

Example of transferring a XML file using "Wget":

```
Using: wget [options] [URL]

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

Parameter	Explanation
--post-file=	Use of the POST method to send a file
file.xml	File that contains the data to be transferred. Details on the structure of the file are provided in "Structure of the XML file" on page 181.
--output-document=	OPTIONAL The answer from the server should be saved in the specified file.
response.txt	OPTIONAL Name of the file in which the answer from the server should be saved. If the data could be transferred successfully, the file contains the following entry: "`+OK: n Records processed`", whereby "n" stands for the number of accepted entries.
--no-check-certificate	***Important note:*** *Deactivates the certificates being checked. Not recommended for continuous operation. Only for test purposes.* "Wget" is not able to access the directory in which Windows saves the certificates. These must be copied into the "Wget" program folder. Information on this is provided in the "Wget" documentation.
"https://...."	HTTP POST command. For details, see "Configuration of a HTTP-POST command" on page 180.

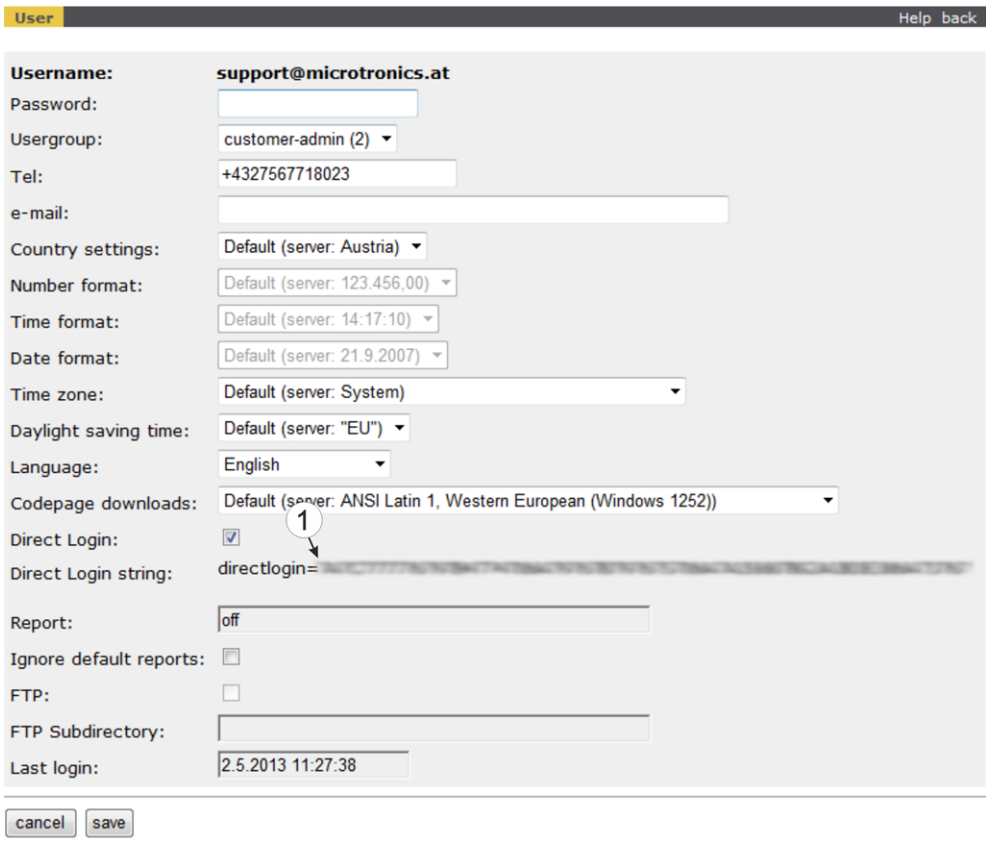
10.4.1.1 Configuration of a HTTP-POST command

> ***Important note:*** *The transmission of XML files to the server is completed via the HTTP server (standard port 80) and not the xPath server (standard port 8080).*

The HTTP-POST command is made up of the following:

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

The XML file must be transferred with this POST command. The configuration of these is described in the chapter "Structure of the XML file" on page 181.

Parameter	Explanation
{URL}	Address of the server to which the data should be transferred (Cloud server)
{Port}	OPTIONAL Port on which the HTTP server is running. Standard value is 80.
{login}	The direct login parameter clearly identifies the user on the server. Each relevant user can only request his own parameter. For this purpose, the user must open the input screen for editing his user data (see "myDatanet Server Manual " 805002) in the "User" area of the Cloud server. The "Direct login character string" (1) is in the lower third of this area. 

10.4.2 Structure of the XML file

10.4.2.1 Changing existing data

If an existing data record has to be changed, the unique identification number of the data record that is going be changed must be specified as the `id` attribute (e.g. `id ="F000002200000282"`). The XML file must only include those elements that should be changed. All of the elements that are not included are not changed. The attributes (`title` , `units` , `calc` , etc.) are not relevant for the data import and can be omitted.

Note: *Formats (comma, date format, etc.) are adopted from the user or server settings.*

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

rm2mraw
: *Not relevant for the data import. However, it is recommended to use the relevant table identifier.*

id
: *Unique identification number of the data record that is going to be modified*

data
: *Value to which the data should be changed*

10.4.2.2 Importing data

The `id="0"` attribute must be specified if a new data record is going be created. The time stamp for the data record can be determined using the `stamp` attribute. It must be ensured that the time stamp of the data is always in ascending order. The `stamp` attribute can also be omitted. In this case, the current time is used as the time stamp for the data record. The default values of the Cloud server are used for all of the elements that are not specified in the XML file when creating a new data record. The `tag` element is a mandatory field and specifies the table in which the data record is saved. The link between the data record and the site (if the data record is saved in one of the tables assigned to the site) or the device (if the data record is saved in one of the tables assigned to the device) is specified via the `context` element. This relates to the unique data record identification number (`id`) of the site or device. The attributes (`title` , `units` , `calc` , etc.) are not relevant for the data import and can be omitted.

Note: *Formats (comma, date format, etc.) are adopted from the user or server settings.*

Site to which the data record should be linked:

```
<Result>
  <rm2mid  id="A1526911CF9CCAA0"  stamp="13.2.2013 10:00:54" >
    <tag> FFFF6700 </tag>
    <name title="Name"> Site1 </name>
    :
    :
  </rm2mid >
</Result>
```

XML file:

```
<?xml version="1.0"?>
<Result>
  <rm2mraw  id="0" stamp="16.1.2013 09:00">
    <data>
        0000003B188A5F8541BF800044170000000000000000000000000000
        00000000000000000000000000000000000000000000000000000000
        00000000000000000000000000000000000000000000000000000000
        00000000000000000000000000000000000000000000000000000000
        000000000000000000000000000000000000000000
    </data>
    <tag>00006700</tag>
    <context>A1526911CF9CCAA0</context>
    </rm2mraw >
</Result>
```

rm2mraw

Not relevant for the data import. However, it is recommended to use the relevant table identifier.

id

`id="0"` *must always be specified when importing data. A new data record with an automatically generated ID is created.*

stamp

OPTIONAL

Time stamp that should be used for the data record. It must be ensured that the time stamp of the data is always in ascending order. If the attribute is omitted, the current time is used as the time stamp for the data record.

data

The actual measurement data that should be imported.

tag

Specifies the table in which the data record is saved

context

Specifies with which site or with which device the data record is to be linked (see "Site with which the data record should be linked")

6872
6872
6872

Chapter 11 Placing into operation

11.1 Placing the system into operation

1. Log into the Cloud server. If you do not already have an account on the Cloud server, you must first create an account (see "New registration and logging in" on page 115).
2. Create a site for the operation on the Cloud server (see "Creating a new site" on page 123).
3. Complete all of the steps detailed in the chapter "Connecting the sensors, actuators and power supply" on page 93. The rapidM2M module M120 is then ready for operation and should have already completed the first connection to the Cloud server.
4. Load the required script into the rapidM2M module M120 via the rapidM2M Toolset (see ""Script" tab" on page 144).

   Important note: Before you import a script into the rapidM2M module M120 via the USB interface, ensure that a site for the operation on the Cloud server has been created and that the device has already completed one synchronisation with the site (see "Testing communication with the device" on page 185). Otherwise the script imported via the USB is deleted during the first synchronisation with the site.

The following step is not mandatory.

5. Check whether the connection to the Cloud server and the transfer of the data has worked correctly (see "Testing communication with the device" on page 185).

11.2 Testing communication with the device

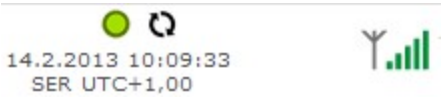
1. Create a site for the operation on the Cloud server (see "Creating a new site" on page 123).
2. Either supply the rapidM2M board for developers via the DC coaxial connector or terminal strip (see "Connecting the sensors, actuators and power supply" on page 93).

   If the jumper for switching on the wireless GSM controller (J37 "MODULE ON") is closed (delivery status, see "Overview" on page 67), the controller of the M120 module starts up once the energy supply of the developer board has been established and as soon as it has detected the "Module_ON" signal (approx. 1,5sec. after the power supply is generated). The status LED should then light up for approx. 1sec. to indicate the boot process (see "Status LED (LED100)" on page 79). The rapidM2M developer package M120 then logs itself into the network and establishes a connection to the Cloud server.

The following step is only necessary, if the jumper for switching on the wireless GSM controller (J37 "MODULE ON") is not closed.

3. Then press the button to switch on the wireless GSM controller S101 (see "Button to activate the wireless GSM controller on the rapidM2M module M120 (S101)" on page 80). The status LED should then light up for approx. 1sec. to indicate the boot process (see "Status LED (LED100)" on page 79). The rapidM2M developer package M120 then logs itself into the network and establishes a connection to the Cloud server.

4. Wait until the measurement instrument list (see "Sites" on page 121) 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 M120 module 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.

Alternatively, the status indication of the rapidM2M Toolset (see "Overview" on page 142) can be evaluated.



In both cases, also evaluate the GSM signal strength (antenna symbol with bars indicating the signal strength).

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

5. Load the required script into the rapidM2M module M120 via the rapidM2M Toolset (see ""Script" tab" on page 144).

   Important note: *Before you import a script into the rapidM2M module M120 via the USB interface, ensure that a site for the operation on the Cloud server has been created and that the device has already completed one synchronisation with the site. Otherwise the script imported via the USB is deleted during the first synchronisation with the site.*

6. Then connect the sensors (see "Connecting the sensors, actuators and power supply" on page 93).

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

7. There are three options available for testing the data transmission:
   - Using the rapidM2M Toolset via the "Server" tab (see ""Server" tab" on page 149)
   - With the help of the "Data Streaming Sample" example web application that is designed to provide the customer with instructions on creating his own user interface (see http://support.rapidm2m.com/lib/test.php tbd):
   - Via the Cloud server using the "Reports" (see "Reports" on page 119). The configuration of the myDatanet Connector (see "myDatanet Connector " on page 277) is required for this purpose.
8. Once you have completed the preparations required for the option that you have selected, initiate a transmission directly on the device if you have included this in your PAWN script. If you have not included an option to initiate a transmission, wait for the next scheduled data transmission.
9. Evaluate the incoming data.

6872
6872
6872

Chapter 12 User interface on the Cloud-Server

12.1 Site configuration

> ***Note:*** *Several of the configuration fields in the following sub chapters may possibly be hidden depending on the respective user level.*

Click on the name of the site in the list of sites to get to the input screen for configuring the site (see "Sites" on page 121).

12.1.1 Site

Customer
Specifies which account the site is assigned to

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!)

12.1.2 Comments

Comments
Free comment field (is also displayed below the device type in the site list)

12.1.3 Control

> ***Note:*** *This configuration section is visible if this site was created using an application template (see "Creating a new site" on page 123).*

Script type	off	Script deactivated	
	Pawn	Activates script processing and informs the M120 module that the script is a "Pawn script".	
Script source	Script	Script	Input window for editing the script that is loaded in M120 module (see "Pawn script" on page 193)
	Upload a compiled script	File upload	Selection of the script binary file (*.amx) that is uploaded to the Cloud server and is loaded into the M120 module 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 structure	Input window for configuring the myDatanet Connector (see "myDatanet Connector " on page 277)		

12.1.4 Configuration 0 - configuration 1

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 Cloud server. For this purpose, the logical structure of the configuration data blocks must be defined with the help of the myDatanet Connector (see "myDatanet Connector " on page 277).

12.1.5 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 volumes	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 KB.

12.1.6 Basic setting

Time zone	Regional settings (not relevant for raw measurement data as this is stored in the UTC)	
Daylight saving time	Configuration for automatic time adjustment	
	Standard	The configuration for the time adjustment is adopted by the global server setting(see tbd).
	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	

12.1.7 FTP export settings

Note: *This configuration section is only visible if the "FTP Agent Extended" licence for the Cloud-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 Cloud 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]	

12.2 Device configuration

> ***Note:*** *Several of the configuration fields in the following sub chapters may possibly be hidden depending on the respective user level.*

You can reach the input screen for configuring the device by clicking on the serial number in the site list (see "Sites" on page 121).

12.2.1 Comments

Comments

Free comment field (is also displayed below the site name in the measurement instrument list)

12.2.2 Measurement instrument

Customer	Name of the customer to whom the measurement instrument is assigned
Serial number	Serial number of the instrument
Device class	The instrument class of the site and instrument must match for an instrument to be able to be connected to a site.
Firmware version	Current software version installed on the measurement controller
Modem version	Current software version installed on the modem controller
OS version	OS version of the modem

Last connection	In each case, the last time stamp of the affected operation	
Last wakeup		
Last disconnection		
Last transmission error		
Last Aloha connection		
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).
Hardware version	Hardware version of the M120 module inserted in the rapidM2M board for developers	

Chapter 13 Pawn script

13.1 General

The following chapter describes the functionality of the pawn script. PAWN (previously SMALL) is a C-similar scripting language that is used on embedded systems. Further information is provided in http://en.wikipedia.org/wiki/Pawn_(programming_language).

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

There are several ways to create a pawn script for the M120 module:

- Direct entry in the "Script" input field in the "Control" configuration section
- Upload of a previously created binary file (*.amx) to the Cloud server
- Use of the script editor of the rapidM2M Toolset

13.1.1 Direct input of a pawn script

The pawn script is entered via the "Control" configuration section (see "Control" on page 189) of the input screen for configuring the site. "Pawn" must be selected as the "Script type" so that the M120 module interprets the commands entered in the "Script" as a pawn script.

13.1.2 Uploading a binary file

If the "Upload a compiled script" entry was selected via the "Script source" list selection in the "Control" configuration section (see "Control" on page 189) of the input screen for configuring the site, a binary file that was, for example, previously created with the rapidM2M Toolset (see ""Script" tab" on page 144) can be uploaded to the Cloud server. This is then loaded into the M120 module during the next connection. When using this method, "Pawn" must also be selected as the "Script type" so that the M120 module interprets the commands as a pawn script.

13.1.3 Script editor of the rapidM2M Toolset

The "Script" tab of the rapidM2M Toolset (see ""Script" tab" on page 144) includes an editor to create and compile prawn scripts. The functional scope of the rapidM2M Toolset includes the transfer of compiled scripts to the M120 module via the USB connection.

13.2 rapidM2M API

13.2.1 Core functions

native heapspace();

Supplies the free memory capacity to the heap

	Explanation
Return value	*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.*

native funcidx(const name[]);

Supplies the index of a public function

Parameter	*Explanation*
name	*Name of the public function*

	Explanation
Return value	• *-1, if there is no function with the transferred name* • *Index of the public function*

native numargs();

Supplies the number of arguments transferred to a function

	Explanation
Return value	*The number of arguments that have been transferred to a function. numargs is useful within functions with a variable list of arguments.*

native getarg(arg, index=0);

Supplies the value of the argument

Parameter	*Explanation*
arg	*The sequence number of the argument. Use 0 for the first argument.*
index	*Index if "arg" refers to an array*

	Explanation
Return value	*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. The return value is the value of the argument.*

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

Sets the value of the argument

Parameter	*Explanation*
arg	*The sequence number of the argument. Use 0 for the first argument.*
index	*Index if "arg" refers to an array*
value	*Value to which the argument should be set*

	Explanation
Return value	• *true, if the value could be set* • *false, if the argument or index are invalid* *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.*

native tolower(c);

Converts a character into lower case

Parameter	*Explanation*
c	*Character that should be converted to lower case*

	Explanation
Return value	*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.*

native toupper(c);

Converts a character into upper case

Parameter	*Explanation*
c	*Character that should be converted to upper case*

	Explanation
Return value	*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.*

native swapchars(c);

Swaps the order of the bytes

Parameter	*Explanation*
c	*Value for which the bytes should be swapped over*

	Explanation
Return value	*Value for which the bytes in parameter "c" are swapped over (the lowest byte becomes the highest byte)*

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

Parameter	*Explanation*
value1	*Two values of which the larger one is to be determined*
value1	

	Explanation
Return value	*The larger of the two transferred values*

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

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

Parameter	*Explanation*
value	*Value that is to be checked*
min	*Lower limit*
max	*Upper limit*

	Explanation
Return value	• *"value" if the value is between "min" and "max"* • *"min" is the value is less than "min"* • *"max", if the value is greater than "max"*

13.2.2 Console functions

native print(const string[]);

Prints the specified string to the standard output

Parameter	*Explanation*
string	*The character string to be issued. This can include escape sequences.*

	Explanation
Return value	*OK*

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.

Parameter	*Explanation*
format[]	*The format character string to be used*

	Explanation
Return value	• *Number of printed characters* • *ERROR, if not successful*

13.2.3 rapidM2M functions

13.2.3.1 Arrays with symbolic indices

TrM2M_Id

Information for identifying the module

```
// string     rapidM2M module identification (e.g. "rapidM2M M1 HW1.1")
// module     rapidM2M module type (e.g. "M1")
// 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}             ]
```

TrM2M_GSMPos

Information regarding a GSM 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_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 connection
//            0, if no connection has been established yet
// mnc        MNC (Mobile Network Code) of the network that was used
//            for the last connection
//            0, if no connection has been established yet
// simstate   Current SIM state (see "SIM state" in chapter
//            "Constants" on page 200)

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

13.2.3.2 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
```

SIM state

```
//Connection can be initiated via script
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 script
RM2M_SIM_STATE_COLD       = 3,   //End of the contract or fair use policy
violated
RM2M_SIM_STATE_DISCARDED  = 4,   //Device has been decommissioned
```

Setting the signal direction of the GPIOs

```
RM2M_GPIO_INPUT    = 0,          // Input
RM2M_GPIO_OUTPUT   = 1,          // Output
```

Signal levels of the GPIOs

```
RM2M_GPIO_LOW      = 0,          // "low" signal level
RM2M_GPIO_HIGH     = 1,          // "high" signal level
```

Configuration of clock polarity and clock phase of the SPI interface

Configuration flag for the rM2M_SpiInit() function

```
RM2M_SPI_CLKPOL    = 0b00000001, // Clock polarity: Idle "high"
RM2M_SPI_CLKPHA    = 0b00000010, /* Clock phase: Data are adopted at the first
                                    edge. */
```

Configuration of the UART interface

Configuration flags for the rM2M_UartInit() function

```
RM2M_UART_1_STOPBIT   = 0b00000001, // 1 stop bit
RM2M_UART_2_STOPBIT   = 0b00000010, // 2 stop bit
RM2M_UART_PARITY_NONE = 0b00000000, // no parity
RM2M_UART_PARITY_ODD  = 0b00000100, // odd parity
RM2M_UART_PARITY_EVEN = 0b00001000, // even parity
RM2M_UART_7_DATABIT   = 0b00000000, // 7 data bits
RM2M_UART_8_DATABIT   = 0b00010000, // 8 data bits
RM2M_UART_FLOW_NONE   = 0b00000000, // no flow control
RM2M_UART_FLOW_RTSCTS = 0b01000000, // RTS/CTS hand shake
```

Selection of the edge that should initiate the interrupt

Configuration flags for the rM2M_IrqInit() function

```
RM2M_IRQ_RISING    = 0,          // with rising edges
RM2M_IRQ_FALLING   = 1,          // with falling edges
```

Connection flags

Control flags for the rM2M_TxStart() function

```
RM2M_TX_POSUPDATE  = 0b00000001, /* Update of the GSM position data when
                                    establishing a connection */
```

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
```

Connection status

Return values of the rM2M_TxGetStatus() function

```
RM2M_TX_FAILED     = 0b00000001, // Connection failed
RM2M_TX_ACTIVE     = 0b00000010, // GPRS connection established
RM2M_TX_STARTED    = 0b00000100, // Connection started
RM2M_TX_RETRY      = 0b00001000, // Waiting time until retry
RM2M_TX_WAKEUPABLE = 0b00010000, // Modem is logged into the GSM network
```

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
```

Configuration flags for the rM2M_CfgInit() function

```
RM2MCFG_VOLATILE   = 0b00000001, // volatile storage (RAM)
```

Indices of the registration memory blocks

that can be accessed via the "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()" and "rM2M_RegSetValue()" functions. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 62.

```
//System-specific data
RM2M_REG_SYS_OTP   = 0, // once during production (recommended)
RM2M_REG_SYS_FLASH = 1, // can be changed during operation (not via script)

//Application specific data
RM2M_REG_APP_OTP   = 2, // once during production (recommended)
RM2M_REG_APP_FLASH = 3, // can be changed during operation (also via script)

//Number of registration memory blocks
RM2M_REG_NUM_REGS  = 4,
```

NMEA error codes

Error codes of the rM2M_SetPosNMEA() function

```
RM2M_NMEA_ERR_DATATYPE  = -2, // Data type (e.g. $GGSA) 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.
```

13.2.3.3 Callback functions

public func(const data{}, len);

Function to be provided by the script developer, that is called up when characters are received via the UART interface.

Parameter	*Explanation*
data	*Array that contains the received data*
len	*Number of received bytes*

public func(pinstate);

Function to be provided by the script developer, that is called up in the event of an interrupt at one of the interruptible pins.

Parameter	*Explanation*
pinstate	*Signal level at the interruptible pin* • *RM2M_GPIO_LOW for "low" at the interruptible pin* • *RM2M_GPIO_HIGH for "high" at the interruptible pin*

public func(cfg);

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

Parameter	*Explanation*
cfg	*Number of the changed configuration memory block starting with 0 for the first memory block*

public func(reg);

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

Parameter	*Explanation*
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 200) that has been changed*

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

Function to be provided by the script developer, that is called up if an SMS is received

Parameter	*Explanation*
SmsTel	*String that contains the telephone number of the sender of the SMS*
SmsText	*String that contains the content of the SMS*

13.2.3.4 Functions

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

Provides the information to identify the module

Parameter	*Explanation*
id	*Structure for storing the information to identify the module (see "TrM2M_Id" in chapter "Arrays with symbolic indices" on page 198)*
len	*Size (in cells) of the structure to store the information - OPTIONAL*

	Explanation
Return value	• *Size (in cells) of the structure used to store the information* • *ERROR if the address and/or length of the ID structure are invalid (outside the script data memory)* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)* ***Note:*** *The firmware of a module detects if a script 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.*

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.

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

	Explanation
Return value	• *timestamp = 0: Seconds since 31.12.1999 (current system time in UTC)* • *timestamp › 0: The transferred time stamp is returned.*

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.

Parameter	*Explanation*
year	*Variable to store the year - OPTIONAL* ***Note:*** *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.)*

	Explanation
Return value	• *timestamp = 0: Seconds since 31.12.1999 (current system time in UTC)* • *timestamp › 0: The transferred time stamp is returned.*

native rM2M_GetTimezoneOffset();

Returns the difference (in seconds) between the system time (UTC) and local time configured for the site on the Cloud 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 Cloud 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.

	Explanation
Return value	*Offset value in seconds*

native rM2M_DoW(timestamp);

Calculates the weekday from a given timestamp

Parameter	*Explanation*
timestamp	*Timestamp of the day in question*

	Explanation
Return value	*Weekday, 0=Monday ... 6=Sunday*

native rM2M_TimerAdd(funcidx);

Generates a new 1s timer

Parameter	*Explanation*
funcidx	*Public function index that should be called up following expiry of the timer* *Type of function: public func();*

	Explanation
Return value	• *OK, if successful* • *ERROR if no valid index was transferred, in the event of an internal error or if no further timers can be created (maximum number reached)* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_TimerRemove(funcidx);

Removes a 1s timer

Parameter	*Explanation*
funcidx	*Public function index of the timer that should be removed* *Type of function: public func();*

	Explanation
Return value	• *OK, if successful* • *ERROR, if no valid index was transferred or in the event of an internal error* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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

Generates a new ms timer

Parameter	*Explanation*
funcidx	*Public function index that should be called up following expiry of the timer* *Type of function: public func();*
cyclic	*Setting for the behaviour following expiry of the timer interval:* *true: The timer must be restarted following expiry of the interval.* *false: The timer is stopped following expiry of the interval.*
time	*Timer interval in milliseconds*

	Explanation
Return value	• *OK, if successful* • *ERROR if no valid index was transferred, in the event of an internal error or if no further timers can be created (maximum number reached)* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_TimerRemoveExt(funcidx);

Removes a ms timer

Parameter	*Explanation*
funcidx	*Public function index of the timer that should be removed* *Type of function: public func();*

	Explanation
Return value	• *OK, if successful* • *ERROR, if no valid index was transferred or in the event of an internal error* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_GpioDir(gpio, dir);

Sets the signal direction for a GPIO

Parameter	*Explanation*
gpio	*Number of the GPIO, starting with 0 for GPIO1*
dir	*Setting the signal direction:* *RM2M_GPIO_INPUT : Input* *RM2M_GPIO_OUTPUT : Output*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_GpioSet(gpio, level);

Sets the output level of a GPIO that was configured as an output

Parameter	*Explanation*
gpio	*Number of the GPIO, starting with 0 for GPIO1*
dir	*Specification of the level to be issued:* *RM2M_GPIO_LOW : Output level set to "low"* *RM2M_GPIO_HIGH : Output level set to "high"*

	Explanation
Return value	• *OK, if successful* • *ERROR_BAD_STATE if the relevant GPIO was not configured as an output* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_GpioGet(gpio);

Reads the signal level of a GPIO that is configured as an input

Parameter	*Explanation*
gpio	*Number of the GPIO, starting with 0 for GPIO1*

	Explanation
Return value	• *RM2M_GPIO_LOW for "low" at the input* • *RM2M_GPIO_HIGH for "high" at the input* • *ERROR_BAD_STATE if the relevant GPIO was not configured as an input* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

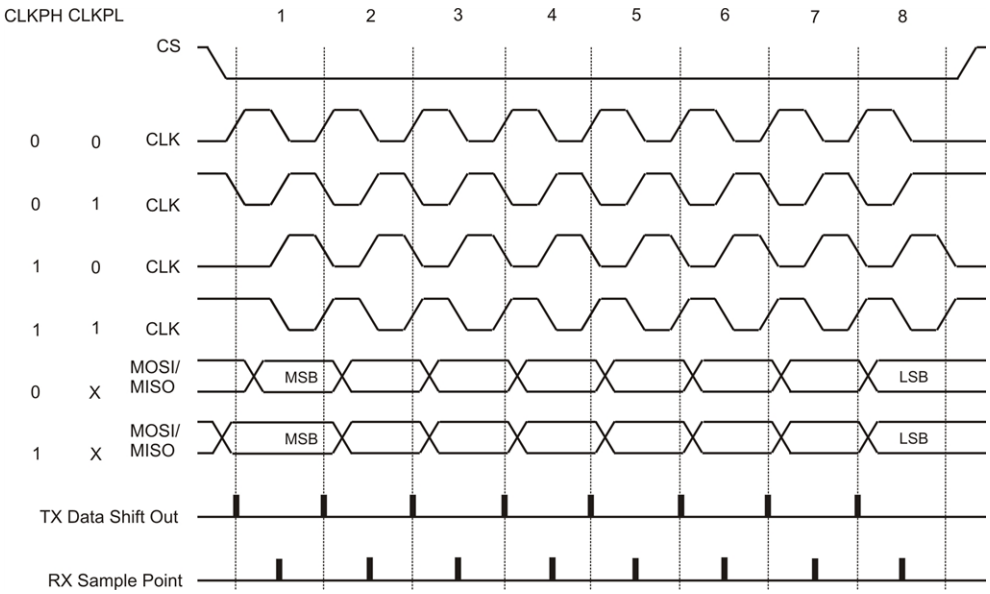
native rM2M_SpiInit(spi, clock, config);

Initialises the SPI interface

Parameter	*Explanation*
spi	*Number of the SPI interface, starting with 0 for SPI1*
clock	*Clock frequency in Hz to be used. Please observe the valid limits for the module being used (see "SPI" on page 109).*
config	*Configuration of clock polarity and clock phase* *Bit0: Clock polarity* *0 = idle "low"* *1 = idle "high"* *Bit1: Clock phase* *0 = Data are issued at the first edge.* *1 = Data are adopted at the first edge.* ***Note:** You can also use the predefined constants for this parameter (see "Configuration of clock polarity and clock phase of the SPI interface" in chapter "Constants" on page 200). The constants can also be combined using the "or" link.*

	Explanation
Return value	• *OK, if successful* • *ERROR, if the number of the SPI interface is invalid* • *ERROR_ALREADY_SUBSCRIBED, if the SPI interface has already been initialised* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

***Note:** Additional information on configuring the clock polarity and clock phase*



native rM2M_SpiClose(spi);

Closes the SPI interface

Parameter	*Explanation*
spi	*Number of the SPI interface, starting with 0 for SPI1*

	Explanation
Return value	• *OK, if successful* • *ERROR, if the number of the SPI interface is invalid* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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

Executes an asynchronous SPI communication. Data is first of all sent and data is then received.

Parameter	*Explanation*
spi	*Number of the SPI interface, starting with 0 for SPI1*
data	*Array in which the data to be sent must initially be saved. Once the data has been sent, the array is used as a memory for the data to be received.*
txlen	*Number of bytes to be sent*
rxlen	*Number of bytes to be received*

	Explanation

Return value	• *OK, if successful*
	• *ERROR, if one of the following errors occurs*
	  ◦ *Number of bytes to be sent ›255*
	  ◦ *Number of bytes to be received ›255*
	• *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_I2cInit(i2c, clock, config);

Initialises the I²C interface

Parameter	*Explanation*
i2c	*Number of the I²C interface, starting with 0 for I2C1*
clock	*Clock frequency in Hz to be used. Please observe the valid limits for the module being used (see "I2C" on page 110).*
config	*Reserved for extensions*

	Explanation
Return value	• *OK, if successful* • *ERROR, if the number of the I²C interface is invalid* • *ERROR_ALREADY_SUBSCRIBED, if the I²C interface has already been initialised* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_I2cClose(i2c);

Closes the I²C interface

Parameter	*Explanation*
i2c	*Number of the I²C interface, starting with 0 for I2C1*

	Explanation
Return value	• *OK, if successful* • *ERROR, if the number of the I²C interface is invalid* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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

Executes an I²C communication. Data is first of all sent and data is then received.

Parameter	*Explanation*
i2c	*Number of the I²C interface, starting with 0 for I2C1*
adr	*Address of the I²C slave (Bit7-Bit1, Bit0 unused)*
data	*Array in which the data to be sent must initially be saved. Once the data has been sent, the array is used as a memory for the data to be received.*
txlen	*Number of bytes to be sent*
rxlen	*Number of bytes to be received*

	Explanation

Return value	• *OK, if successful*
	• *ERROR, if one of the following errors occurs*
	  ◦ *Number of bytes to be sent >255*
	  ◦ *Number of bytes to be received >255*
	• *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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

Initialises the UART interface

Parameter	*Explanation*
uart	*Number of the UART interface, starting with 0 for UART1*
baudrate	*Baud rate to be used. Please observe the valid limits for the module being used (see "UART" on page 108).*
mode	*Bit 0...1* *1 = 1 stop bit* *2 = 2 stop bit* *Bit 2...3* *0 = no parity* *1 = uneven parity* *2 = even parity* *Bit 4...5* *0 = 7 data bits* *1 = 8 data bits* *Bit 6...7* *0 = no flow control* *1 = RTS/CTS handshake* ***Note:** You can also use the predefined constants for this parameter (see "Configuration of UART interface" in chapter "Constants" on page 200). The constants can also be combined using the "or" link.*
funcidx	*Public function index for the UART character receipt* *Type of function: public func(const data{}, len);*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_UartClose(uart);

Closes the UART interface

Parameter	*Explanation*
uart	*Number of the UART interface, starting with 0 for UART1*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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

Sends data via the specified UART interface

Parameter	*Explanation*
uart	*Number of the UART interface, starting with 0 for UART1*
data	*Array that contains the data to be sent*
len	*Number of bytes to be sent*

	Explanation
Return value	• *OK, if successful* • *ERROR if the number of bytes to be sent >256* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_IrqInit(irq, config, funcidx);

Activates the interrupt functionality of an interruptible pin

Parameter	*Explanation*
irq	*Number of the interruptible pin, starting with 0 for INT0*
config	*Selection of the edge that should initiate the interrupt* *RM2M_IRQ_RISING : with rising edges* *RM2M_IRQ_FALLING : with falling edges*
funcidx	*Public function index that should be executed in the event of an interrupt* *Type of function: public func(pinstate);*

	Explanation

Return value	• *OK, if successful*
	• *ERROR_ALREADY_SUBSCRIBED, if the interrupt functionality for the relevant pin has already been initialised*
	• *ERROR, if one of the following errors occurs*
	  • *Interrupt functionality for this pin not available*
	  • *The function that must be executed in the event of an interrupt is not public.*
	• *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_IrqClose(irq);

Deactivates the interrupt functionality of an interruptible pin

Parameter	*Explanation*
irq	*Number of the interruptible pin, starting with 0 for INT0*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_TxStart(flags=0);

Initiates a connection to the server

Parameter	*Explanation*
flags	*If Bito (RM2M_TX_POSUPDATE) is set, the GSM position data is also updated (see tbd).*

	Explanation
Return value	• *OK, if successful* • *ERROR_SIM_STATE, if a connection is not possible due to the current SIM state (see "SIM state" in chapter "Constants" on page 200)* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_TxSetMode(mode);

Sets the connection type to be used

Parameter	*Explanation*
mode	*Connection type to be used:* *RM2M_TXMODE_TRIG: The connection is established when the "rM2M_TxStart()" function is called up* *RM2M_TXMODE_WAKEUP: The connection is established in the same way as in "Interval" mode when the "rM2M_TxStart()" function is called up. Additionally, the device can be initiated via the server to immediately establish a connection (see "Sites" on page 121). For this purpose, the device immediately logs into the GSM network as soon as this mode it set.*  *RM2M_TXMODE_ONLINE: The device does not separate the connection and continuously transmits the measurement data.The connection is established as soon as this mode has been set. Calling up the "rM2M_TxStart()" function is not necessary.*

	Explanation
Return value	• *OK, if successful* • *ERROR_SIM_STATE, if the mode is not possible due to the current SIM state (see "SIM state" in chapter "Constants" on page 200)* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

Note: *Additional explanation about the connection types*

Connection type	*Energy consumption*	*Data volumes*	*Response time*
Online			
Interval & wakeup			
Interval			

native rM2M_TxGetStatus();

Supplies the current connection status

	Explanation	
Return value	*Bito (RM2M_TX_FAILED):*	*set if the last GPRS connection establishment failed*
	Bit1 (RM2M_TX_ACTIVE):	*set when a GPRS connection is established*
	Bit2 (RM2M_TX_STARTED):	*set when a connection establishment has been started*
	Bit3 (RM2M_TX_RETRY):	*set during the waiting time until the next automatic retry in the event of connection problems*
	Bit4 (RM2M_TX_WAKEUPABLE):	*set when the modem is logged into the GSM network (wakeup possible)*

native rM2M_GSMGetRSSI();

Supplies the GSM signal strength

	Explanation
Return value	*Signal strength in [dBm] (maximum value range: -128...127)* *GSM values range from -113 to -51 dBm.*

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.

Parameter	*Explanation*
timestamp	*Time stamp that should be used for the recording* *= 0: The current system time is used as the time stamp.* *› 0: The transferred time stamp is used.* *(The time stamp must be specified in seconds since 31.12.1999)*
data	*Array that comprises the data to be saved*
len	*Number of bytes to be saved (max. 1024 Byte)*

	Explanation

Return value	• *OK, if successful*
	• *-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.*
	• *ERROR, if one of the following errors occurs*
	  • *Memory area (data{}, len) is invalid.*
	  • *More than 10 calls during one script run.*
	  • *Number of bytes to be saved › 1024 Byte*
	  • *FLASH write process not successful*
	  • *The time stamp transfer parameter is more than 5 minutes in the future*
	• *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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

Writes the transferred value to a specified position in an array

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

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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.

	Big endian			*Little endian*		
Addresses	*Hex*	*Dez*	*Binary*	*Hex*	*Dez*	*Binary*
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

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

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

Parameter	*Explanation*
data	*Array that should be used as a data area for a data record or a configuration*
pos	*Byte offset within the array to determine the position from which the data should be read*
value	*Variable to store the data to be read*
size	*Number of bytes that should be read*
bigendian	*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.*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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.

	Big endian			*Little endian*		
Addresses	*Hex*	*Dez*	*Binary*	*Hex*	*Dez*	*Binary*
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

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

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

Function to access packed data. If the Bito (RM2M_PACK_GET) of the "type" parameter was set, the function returns the value that is located at the specified position in an array. Otherwise the function writes the transferred value to the specified position in the array.

Parameter	*Explanation*
data	*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*
pos	*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*
value	*Set packed: Value that should be written in the array* *Get packed: Value that should be read from the array*
type	*Configuration flags for the function* *Bito: 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* **Note:** *You can also use the predefined constants for this parameter (see "Configuration flags for the rM2M_Pack() function" in chapter "Constants" on page 200). The constants can also be combined using the "or" link.*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_CfgInit(cfg, flags);

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

Parameter	*Explanation*
cfg	*Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.*
flags	*Configuration flags to be set/deleted* *Bito: Type of storage* *0 (default) = stored in FLASH in non-volatile manner* *RM2MCFG_VOLATILE = saved in RAM in volatile manner*

	Explanation
Return value	• *OK, if successful* • *< OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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 Bito was set (Bito = RM2MCFG_VOLATILE), the configuration memory block is saved in the RAM in a volatile manner when the "rM2M_CfgWrite" function is called up. 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. Please note that the configuration memory block is either saved in the RAM in a volatile manner (Bito = RM2MCFG_VOLATILE) or in the FLASH in a non-volatile manner (Bito = 0, default) depending on the type of storage selected via the "rM2M_CfgInit" function. The function is also informed 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 Cloud server during the next connection.

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

	Explanation
Return value	• *OK, if successful* • *ERROR_MEM, if there is not enough temporary memory (RAM) currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_CfgFlush(cfg);

Saves the configuration memory block for which the number was transferred in the FLASH in a non-volatile manner. Calling up the function is only necessary if "volatile in RAM (Bito = RM2MCFG_VOLATILE)" was selected as the type of storage for the relevant configuration memory block via the "rM2M_CfgInit" function.

Parameter	*Explanation*
cfg	*Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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.

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

	Explanation
Return value	• *Size of the memory used in the configuration memory block* • *ERROR_MEM, if there is not enough temporary memory (RAM) currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_CfgDelete(cfg);

Deletes all of the data of the transferred configuration memory block

Parameter	*Explanation*
cfg	*Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.*

	Explanation
Return value	• *OK, if successful* • *ERROR_MEM, if there is not enough temporary memory (RAM) currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_CfgOnChg(funcidx);

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

Parameter	*Explanation*
funcidx	*Public function index that should be called up if the configuration has changed* *Type of function: public func(cfg);*

	Explanation
Return value	• *OK, if successful* • *ERROR, if no valid index of a public function was transferred* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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 62.

Parameter	*Explanation*
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 200)*
name	*Name of the entry*
string	*Array to store the string to be read*
len	*Size (in cells) of the transferred array to store the string to be read - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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 62.

Parameter	*Explanation*
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 200)*
name	*Name of the entry*
value	*Variable to store the value to be read*
tag	*The integer and floating-point conversion are differentiated by the "tag" of the variables. - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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 62.

Parameter	*Explanation*
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 200)*
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	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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 62.

Parameter	*Explanation*
reg	*Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 200)*
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	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_RegOnChg(funcidx);

Specifies the function that should be called up if one of the registration memory blocks has changed. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 62.

Parameter	*Explanation*
funcidx	*Public function index that should be called up if the registration has changed* *Type of function: public func(reg);*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_SmsInit(funcidx, config);

initialises SMS receipt

Parameter	*Explanation*
funcidx	*Index of the public function that should be called up if an SMS has been received* *Type of function: public func(const SmsTel[], const SmsText[]);*
config	*Reserved for extensions*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native rM2M_SmsClose();

deactivates SMS receipt

	Explanation
Return value	*OK*

native rM2M_WriteLog(log, param);

Generates an entry in the device log

Parameter	*Explanation*
log	*Log entry to be generated (valid range 0...999).* *Log entries are limited to the valid range. This means that 0 is used if it falls below the valid range and 999 is used if the valid range is exceeded.* ***Important note:** The log entries are mapped in the "MODULE ERR" (2000-2999) area for display on the server.*
param	*Additional parameter for the exact specification of the log entry (valid range -32768 .. 32767).*

	Explanation
Return value	*OK*

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 transferred to the Cloud server and can, for example, be read out via the XML interface (see "rm2mpos" on page 163).

Parameter	*Explanation*
Lat	*geographical latitude in degrees (resolution: 0.000001°)* *-90 000 000 = South pole 90° south* *0 = Equator* *+90 000 000 = North pole 90° north*
Long	*geographical longitude in degrees (resolution: 0.000001°)* *-180 000 000 = 180° west* *0 = Zero meridian (Greenwich)* *+180 000 000 = 180° east*
Elev	*Height above sea level in meters (valid range: -999...+9999)*
Qual	*Quality indicator (GPS quality indicator)* *0 = Invalid/no fix* *1 = Non-differential GPS fix* *2 = Differential GPS fix* *3 = Precise positioning service PPS* *4 = Real time kinematic RTK* *5 = Float RTK* *6 = Estimated fix (dead reckoning, coupled navigation)* *7 = Manual input mode* *8 = Simulation mode*
SatUsed	*Number of satellites used for positioning (valid range: 0 ... 99)*

	Explanation
Return value	• *OK, if successful* • *ERROR* ***Note:** The parameters are checked against the specified range limits. The function returns "ERROR" if the limits are not adhered to.*

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 transferred to the Cloud server and can, for example, be read out via the XML interface (see "rm2mpos" on page 163).

Parameter	*Explanation*
Sentence	*NMEA data record from a GPS receiver starting with the '$' character. The following data records are currently supported:* • *$GPGGA - location specification (fix information)* ***Important note:** The strings must be terminated ('\0') immediately after the checksum.*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "NMEA error codes" in chapter "Constants" on page 200)*

native rM2M_GetPos(&Lat, &Long, &Elev);

Reads out the GPS position information saved in the device

Parameter	*Explanation*
Lat	*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*
Long	*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*
Elev	*Variable to store the height above the seal level in meters (valid range: -999...+9999)*

	Explanation
Return value	• *OK if valid GPS position information is stored in the device* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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

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

Parameter	*Explanation*
posidx	*Selection of the information returned by the function* *posidx < 0:* *Read out the number of GSM cells for which valid information is saved in the device* *posidx >=0:* *Number of the GSM cell information block that should be read out*
pos	*posidx < 0:* *Not required* *posidx >=0:* *Structure for storing the information about a GSM cell in the receiving range (see "TrM2M_GSMPos" in chapter "Arrays with symbolic indices" on page 198)*

	Explanation
Return value	*posidx < 0:* *Number of GSM cells for which valid information is saved in the device (max. 10)* *posidx >=0:* • *OK, if the desired GSM cell information block contains valid data* • *ERROR*

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

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

Parameter	*Explanation*
info	*Structure for storing the information (see "TrM2M_GSMInfo" in chapter "Arrays with symbolic indices" on page 198)*
len	*Size (in cells) of the structure to store the information - OPTIONAL*

	Explanation
Return value	• *Size (in cells) of the structure used to store the information* • *ERROR if the address and/or length of the info structure are invalid (outside the script data memory)* • *< OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

13.2.4 File transfer functions

13.2.4.1 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)
// size       File size in byte
// crc        Ethernet CRC32 of the file (see "CRC32()" )
// flags      File flags (see "File flags" in
//            chapter "Constants" on page 236)
#define TFT_Info[  .name{256}, .stamp, .size, .crc, .flags ]
```

13.2.4.2 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    /* files nodes (required to entitle the server to
                              create a new file) */
FT_FLAG_SYSTEM = 0x0008    /* System file (cannot be used by the
                              script) */
```

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.
```

13.2.4.3 Callback functions

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

Function to be provided by the script 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 236).

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*
cmd	*File transfer command that was received from the system and that has to be processed by the callback function*
data	*This parameter is only relevant when the following file transfer commands are received:* • *FT_CMD_STORE: Array that contains the properties of the file that should be newly created. Structure:* • *FT_CMD_WRITE: Array that contains the data received from the Cloud server.*
len	*This parameter is only relevant when the following file transfer commands are received:* • *FT_CMD_READ: Number of bytes requested by the Cloud server* • *FT_CMD_STORE: Size of the file property block received from the Cloud server* • *FT_CMD_WRITE: Number of bytes received from the Cloud server*
ofs	*This parameter is only relevant when the following file transfer commands are received:* • *FT_CMD_READ: Byte offset within the file of the data block to be transferred to the Cloud server* • *FT_CMD_WRITE: Byte offset within the file of the data block received from the Cloud server*

Offset	*Bytes*	*Explanation*
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*

13.2.4.4 Functions

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

Registers a file made available by the PAWN script

Parameter	*Explanation*
name	*Unique file name*
id	*Unique identification with which the file is then referenced (freely selectable)*
funcidx	*Index of the public function that should be called up if a file transfer command has been received* *Type of function: public func(id, cmd, const data{}, len, ofs);* ***Important note:** All file transfer commands (see "File transfer commands" in chapter "Constants" on page 236) must be handled by this public function.*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native FT_Unregister(id);

Removes a file from the registration. The file is no longer available for the file transfer.

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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.*

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*
stamp	*Time stamp of the file (seconds since 31.12.1999)*
size	*File size in byte*
crc	*Ethernet CRC32 of the file (see "CRC32()")*
flags	*File flags (see "File flags" in chapter "Constants" on page 236)*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native FT_Read(id, const data{}, len);

Transmits the data to the system, to transfer it to the Cloud 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.*

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*
data	*Array that contains the data that should be transmitted to the system to be transferred to the Cloud server*
len	*Number of bytes to be transferred*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native FT_Accept(id, newid=-1);

Accepts the file that the Cloud 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 237) must be saved via the "FT_SetProps()" function.

> ***Important note:*** *This function must be called up following receipt of a "FT_CMD_STORE" command.*

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*
newid	*Unique identification for the new file that should be created (-1 if it is not a new file) - OPTIONAL*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native FT_Written(id, len);

Confirms that the data received from the Cloud 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 237).

> ***Important note:*** *This function must be called up following receipt of a "FT_CMD_WRITE" command.*

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*
len	*Number of written bytes*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

native FT_Error(id);

Used to display a file handling error and terminates any file command

Parameter	*Explanation*
id	*Unique identification with which the file is referenced (specified during registration)*

	Explanation
Return value	• *OK, if successful* • *‹ OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

13.2.5 Mathematical functions

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	π
M_PI_2	1.57079632679489661923	π/2
M_PI_4	0.78539816339744830962	π/4
M_1_PI	0.31830988618379067154	1/π
M_2_PI	0.63661977236758134308	2/π
M_2_SQRTPI	1.12837916709551257390	2/sqrt(π)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)

native fround(Float:x);

Commercially rounds the transferred float

Parameter	*Explanation*
x	*Float that should be rounded*

	Explanation
Return value	*Commercially rounded integral value*

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 [-π/2, π/2], x element of [-1, 1]

native Float:acos(Float:x);
Arccosine(x) in the range [0, π], x element of [-1, 1]

native Float:atan(Float:x);
Arctangent(x) in the range [-π/2, π/2]

native Float:atan2(Float:y, Float:x);
Arctangent(y/x) in the range [-π, π]

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 <= 0, or in x < 0 and y is not a whole number.

native Float:sqrt(Float:x);
Root x, x >= 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);
$x*2^n$

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.

13.2.6 String functions

> ***Note:*** *You require the following include file to be able to use the functions in this chapter:*
> `#include <string>`

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

native strlen(const string[]);

Supplies the length of the string (without '\0')

Parameter	*Explanation*
string	*Character string for which the length has to be determined*

	Explanation
Return value	*Number of characters without the final '\0'*

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 "snprintf" function of the standard ANSI-C implementation.

Parameter	*Explanation*
dest	*Array to store the formatted result*
maxlength	*Maximum number of characters that the array dest can store*
format	*The format character string to be used*

	Explanation
Return value	• *-1 in the event of a fault* • *Number of characters that would have been written if the array dest had been long enough (without '\0').* *The array dest is always assigned a final zero. The length of the array dest cannot be exceeded.*

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

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

Parameter	*Explanation*
dest	*Array to store the character string that should be copied*
source	*Character string that should be copied*
maxlength	*Number of characters to be copied - OPTIONAL*

	Explanation
Return value	*Number of copied characters*

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

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

Parameter	*Explanation*
dest	*Array to store the result. This array already comprises 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	*Number of characters to be added - OPTIONAL*

	Explanation
Return value	*Number of added characters*

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

Compares character string1 and string2

Parameter	*Explanation*
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*

	Explanation
Return value	• *1: string1 › string 2* • *0: both of the character strings are the same (at least the length that is taken into account)* • *-1: string1 ‹ string 2*

native strchr(const string[], char);

Searches for a character (first occurrence) in a character string

Parameter	*Explanation*
string	*Character string that should be searched*
char	*Character that the search is looking for*

	Explanation
Return value	• *-1, if the character that the search is looking for is not included in the character string* • *Array index for the character that the search is looking for (first character occurring in the character string)*

native strrchr(const string[], char);

Searches for a character (last occurrence) in a character string

Parameter	*Explanation*
string	*Character string that should be searched*
char	*Character that the search is looking for*

	Explanation
Return value	• *-1, if the character that the search is looking for is not included in the character string* • *Array index for the character that the search is looking for (last character occurring in the character string)*

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)*

Parameter	*Explanation*
string1	*Character string that should be searched*
string2	*Character string of permitted characters*

	Explanation
Return value	• *Length of string1 if no permitted characters are found* • *Position of the first character in the character string that should be searched that is not included in the character string of permitted characters*

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)

Parameter	*Explanation*
string1	*Character string that should be searched*
string2	*Character string of permitted characters*

	Explanation
Return value	• *Length of string1 if no permitted character has been found* • *Position of the first character in the character string that should be searched that is also included in the character string of permitted characters*

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

Parameter	*Explanation*
string1	*Character string that should be searched*
string2	*Character string of permitted characters*

	Explanation
Return value	• *-1, if the character that the search is looking for is not included in the character string* • *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*

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

Searches character string2 in character string1

Parameter	*Explanation*
string1	*Character string that should be searched*
string2	*Character string that the search should be for*

	Explanation
Return value	• *-1, if character string2 that is being searched for is not included in string1* • *Array index where character string2 that is being searched for starts in string1*

native strtol(const string[], base);

Converts a character string into a value

Parameter	*Explanation*
string	*Character string that is to be converted*
base	*Specifies the basis that must be used for the conversion* *2-36: The specified basis is used* *0: 8, 10 or 16 is used as the basis, depending on the character string that must be converted* *Basis 8: with a leading 0* *Basis 16: with 0x or 0X*

	Explanation
Return value	*Value that corresponds to the character string*

native Float: atof(const string[]);

Converts a character string into a float

Parameter	*Explanation*
string	*Character string that is to be converted*

	Explanation
Return value	*Float for which the numerical value corresponds to the character string*

13.2.7 Help functions

13.2.7.1 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]
```

13.2.7.2 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
};
```

13.2.7.3 Functions

native getapilevel();

Issues the implemented API level of the script engine

	Explanation
Return value	*Implemented API level of the script engine*

native CRC16(data{}, len);

Supplies the calculated modbus CRC16 of the transferred data

Parameter	*Explanation*
data	*Array that contains the data for which the CRC16 must be calculated*
len	*Number of bytes that must be taken into consideration during the calculation*

	Explanation
Return value	*Calculated CRC16*

native CRC32(data{}, len);

Supplies the calculated Ethernet CRC32 of the transferred data

Parameter	*Explanation*
data	*Array that contains the data for which the CRC32 must be calculated*
len	*Number of bytes that must be taken into consideration during the calculation*

	Explanation
Return value	*Calculated CRC32*

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

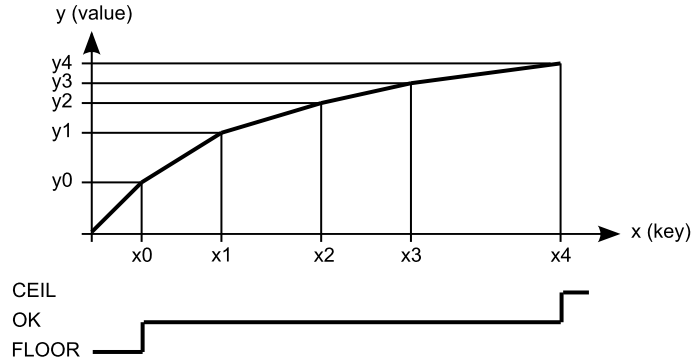
*Searches for a certain value in the "key" column of the transferred reference point table and supplies the relevant value from the "value" column in the table. If the searched value is between two reference points, the returned value is interpolated linearly between the two adjacent values in the "value" 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	• *OK, if the relevant value was found* • *TAB_ERR_FLOOR, if the searched value is lower than the first table entry. "value" contains the first table entry.* • *TAB_ERR_CEIL, if the searched value is higher than the last table entry. "value" contains the last table entry.* • *‹ OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 200)*

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.

13.3 Pawn script error codes

The script execution is stopped and deactivated if an error occurs while the pawn script is being processed. The relevant error code is also entered in the device log. The parameter for all log entries except "SCRIPT_ERR" contains the 32-bit instruction pointer of the PAWN 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-Bito of 32-bit instruction pointer. Instructions on evaluating the device log are included in the chapter "Log tab" (see ""Log" tab" on page 149).

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3000	SCRIPT_ERR	0	NO SCRIPT	No valid script available
		1	SCRIPT UPDATE	New script received
		2	SCRIPT EXCEPT LOOP	Exception loop detected (4 system starts due to exception within 10 min.). The script is deactivated and error handling is activated (see "Error handling" on page 62). 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. Script was restarted.
		4	SCRIPT SOFT ERROR 2	Second time a runtime error has occurred within 24 hours. Script was restarted.
		5	SCRIPT SOFT ERROR 3	Third time a runtime error has occurred within 24 hours. Script was restarted. The script is deactivated and error handling is activated if another runtime error should occur within 24 hours (see "Error handling" on page 62).
		6	SCRIPT UPDATE ERROR	Connection abort during the script download As the existing script 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.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3001	AMX_ERR_EXIT	##	---	Abortion e.g. max. number of PAWN 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 area
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	##	---	Divide 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 area.
3027	AMX_ERR_GENERAL	##	---	General error (invalid or non-specific error)
3028	AMX_ERR_OVERLAY	##	---	Overlays are not supported (JIT) or are not initialised.

13.4 Syntax

13.4.1 General syntax

13.4.1.1 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.

13.4.1.2 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.

13.4.1.3 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.

13.4.1.4 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. Pawn distinguishes 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.

13.4.1.5 Reserved keywords

Statements	Operator	Directives	Others
assert	defined	defined	defined
break	sizeof	sizeof	sizeof
case	state	state	state
continue	tagof	tagof	tagof
default			
do			
else			
exit			
for			
goto			
if			
return			
sleep			
state			
switch			
while			

13.4.1.6 Numerical constants

13.4.1.6.1 Numerical integer constants

Binary:
ob followed by a series of 0 and 1

Decimal:
A series of numbers between 0 and 9

Hexadecimal:
ox followed by a series of numbers between 0 and 9 and the letters a to f

13.4.1.6.2 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.

13.4.2 Variables

13.4.2.1 Declaration

The keyword "new" declares a new variable. For special declarations, the keyword "new" is replaced with "static" (see "Static local declaration" on page 255). 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 266) - local variable

Example:

```
new a;     // without initialisation (value is 0)
new b = 3; // with initialisation (value is 3)
```

13.4.2.2 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.

13.4.2.3 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.

13.4.2.4 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.

13.4.2.5 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.

13.4.2.6 Floating point values

The pawn supports floating point values. 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)
```

13.4.3 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 270.

13.4.4 Array variables

13.4.4.1 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]".

13.4.4.2 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 */
```

13.4.4.3 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
```

13.4.4.4 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 258).

13.4.4.5 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.

13.4.5 Operators and expressions

13.4.5.1 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

13.4.5.2 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
```

13.4.5.3 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 in the pawn. The operators ++ and -- change the operand. The operand must be a "lvalue".*

13.4.5.4 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.

13.4.5.5 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

13.4.5.6 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.

13.4.5.7 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".

13.4.5.8 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.

13.4.5.9 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 array index (element) array index (character)	left-to-right
! ~ - ++ -- : defined sizeof tagof	logical not one's complement two's complement (unary minus) increase decrease "tag" overwritten symbol definition status symbol size in "elements" unique number of the tag	right-to-left
* / %	multiplication division modulo	left-to-right
+ -	addition subtraction	left-to-right
›› ››› ‹‹	arithmetic shift to the right logical shift to the right shift to the left	left-to-right
&	bitwise "and"	left-to-right
^	bitwise "exclusive or"	left-to-right
|	bitwise "or"	left-to-right
‹ ‹= › ›=	less than less or equal to greater than greater or equal to	left-to-right
== !=	equal unequal	left-to-right
&&	logical "and"	left-to-right
||	logical "or"	left-to-right
? :	conditional execution	right-to-left
=	Assignment *= /= %= += -= ››= ›››= ‹‹= &= ^= |=	right-to-left
,	comma	left-to-right

13.4.6 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.

13.4.6.1 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.

13.4.6.2 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".

13.4.6.3 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.

13.4.6.4 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.

13.4.6.5 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
}
```

13.4.6.6 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
}
```

13.4.6.7 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
}
```

13.4.6.8 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
}
```

13.4.6.9 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.

13.4.6.10 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.

13.4.6.11 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).

13.4.6.12 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
}
```

13.4.6.13 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.

13.4.6.14 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
}
```

13.4.6.15 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
}
```

13.4.7 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 258.

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 x^y */
  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.

13.4.7.1 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 */
```

13.4.7.2 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.

13.4.7.3 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]
}
```

13.5 Example

tbd

13.6 Differences to C

- The pawn is missing the input mechanism of C and 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. Values such as ".5" and "6." are acceptable in C but must be written as "0.5" and "6.0" in the pawn. The decimal point is optional in C. If an exponent is included, then you can write "2E8" in C, however the pawn will not accept the capital letter "E". Use the lower case letter "e". It requires a comma: e.g. "2.0e8" (see "Numerical constants" on page 254).
- The pawn does not support any "pointer". The pawn includes a "reference" argument to transfer function parameters as a reference (see "Function arguments ("call-by-value" versus "call-by-reference")" on page 270). The "placeholder" argument replaces some applications of the ZERO pointer (see "Standard values of function arguments" on page 273).
- Numbers can be specified in a hexadecimal, decimal or binary format. The octal format is not supported (see "Numerical constants" on page 254). 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 268). The "switch" statement is a "conditional goto" in C/C++. The "switch" statement is a structured "if" in the pawn.
- A "break" statement only terminates loops. In C/C++, the "break" statement also terminates a "case" in a "switch" statement.
- The pawn supports "array assignments" 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, the pawn also supports literal arrays and thus expressions such as "a = {0,1,2,3,4,5}" where "a" is an array variable with six elements.

- "defined" is an operator and not a preprocessor directive. The "defined" operator in the pawn 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 262.
- An empty statement is an empty block (with {}) and not a semicolon (see "Composite statements" on page 264). 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 258).
- 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" in the pawn implements the "enum" functionality of C.
- In most cases, the forward declarations of functions (i.e. prototypes) are not necessary. A pawn is a two-pass compiler. 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. The pawn attends to the parameter name in the prototype due to the "named parameter" function. The pawn uses prototypes 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.

6872
6872
6872

Chapter 14 myDatanet Connector

The basic principle of the rapidM2M module is "storage-2-storage" data transmission. For this type of data transmission, neither the rapidM2M module 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 rapidM2M module to the sever can therefore be selected freely. There are 1024 Byte available per data record that can be used as required. There are also another 10 independent memory blocks each with 10kB 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 rapidM2M module can also be used within the myDatanet interface (reports, visualisations, graphics, etc.). The myDatanet Connector contains the tool for describing the data as well as the correct provision of the data for use within the interface.

14.1 Prerequisites

> ***Important note:*** *Please read this chapter carefully before using the myDatanet Connector . It is important that these specifications are observed as the correct functionality of the myDatanet Connector can otherwise not be guaranteed.*

Certain prerequisites, that limit the basic dynamics of rapidM2M but that are required to ensure that the data is also available in myDatanet , must be met to be able to use the myDatanet Connector for the rapidM2M .

- Use of a data preamble (only applies to data)

  At the start of a data record to separate the different usage data blocks (e.g. measurement data, alarms, log entries) from one another. It is recommended to use the first one or two bytes for this purpose. The data preamble is used in the ‹split› tags of the data structure description.

  Example for using one byte:

  0x01 ... Measurement data
  0x02 ... Alarms
  0x03 log entries

- Completeness of the usage data block (only applies to data)

  Usage data blocks must always be filled with all of the data fields even if only one or two values have changed since the last transmission. The data is extracted from the usage data block by the myDatanet Connector using the byte offset and length, which is why the fields must always be in the same position.

- Homogeneity of the usage data block (only applies to data)

  A usage data block that is copied to one of the structured measurement data channels ("histdata0" - "histdata9") using the split tag (see "Split tag" on page 278) must always be structured the same way (i.e. must include the same data fields). If different usage data block structures are used, each variant must be assigned its own structured measurement data channel via a split tag.

- Use of big endian (applies to data and configurations)

  In the following example, the whole number 439.041.101 is saved as dword (u32) from memory address 10000.

	Big endian			Little endian		
Addresses	Hex	Dez	Binary	Hex	Dez	Binary
10000	1A	26	00011010	4D	77	01001101
10001	2B	43	00101011	3C	60	00111100
10002	3C	60	00111100	2B	43	00101011
10003	4D	77	01001101	1A	26	00011010

14.2 Split tag

The split tag is used to correctly assign data received from the device on the server.

A rapidM2M module transfers its data in the "rm2mraw" area. This area is only provided for the storage-2-storage use of the server. A split tag must be configured if all of or certain data are to be made available in myDatanet for reports, views, graphics, etc..

> ***Important note:*** *The split tags are not applied to data that is already available on the system. The first byte is evaluated upon receipt of each rapidM2M raw data record. If the value is less or equal to 9, the data record is automatically copied to the respective structured measurement data channel as long as no split tag with different instructions is present. If the first byte contains the value 0, the data record is copied to measurement data channel 0 ("histdata0"). If the first byte contains the value 9, the data record is copied to measurement data channel 9 ("histdata9"). Subsequent processing is not provided.*
>
> ***Important note:*** *If you only want to use the raw data record ("rm2mraw"), you must ensure that the first byte of the raw data record always has a value higher than 9.*

Example of a split tag:

Code	Explanation
`<split>`	Start of the split tag
`source =rm2mraw`	Source of the data that should be copied
`target =histdata0`	Target to which data should be copied
`key =*`	Key for separating the data
`</split>`	End of the split tag

source

Table name for the source
The source is always `rm2mraw` in rapidM2M .

target

Table name for the target
The structured measurement data channels in rapidM2M are designated as "histdata0" - "histdata9".

key

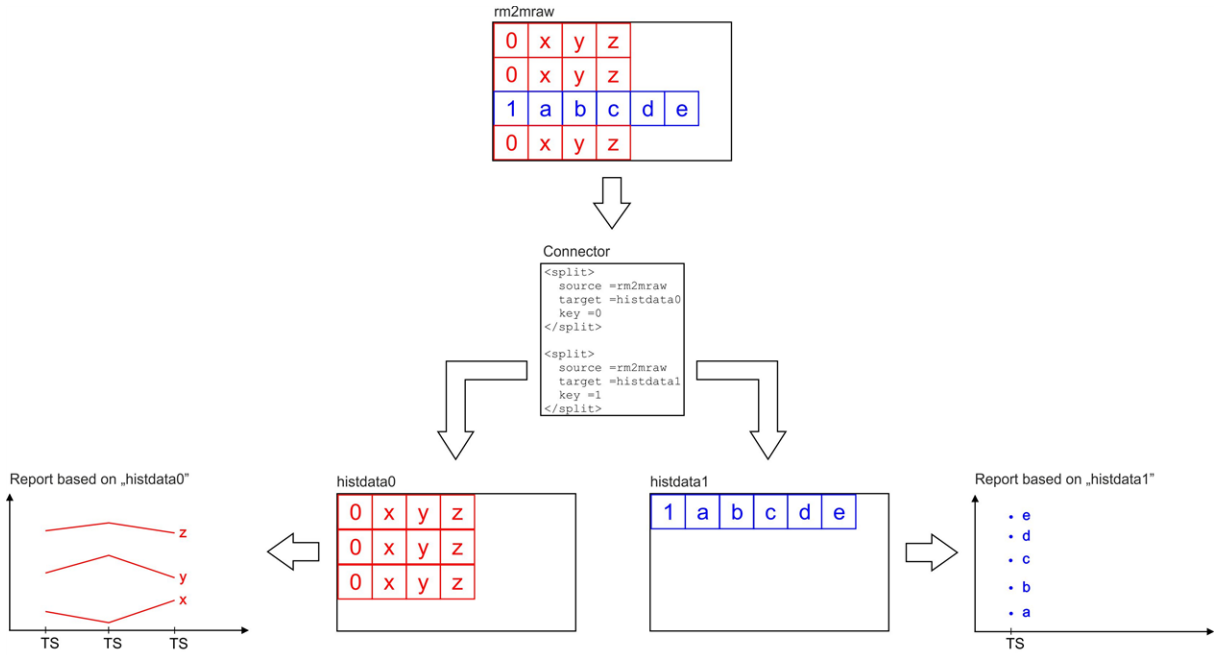
This field provides the option to determine whether the current data record should be copied:

- *All of the data records are copied from the source to the target if a star (*) is used.*
- *By specifying a value in HEX format, a check is completed to determine whether the start of the received data record (also includes more than the first byte if specified accordingly) corresponds to this value. The data record is copied if there is a match.*

Several split tags can be specified. The data record is copied at the first suitable key entry. The following split tags are no longer taken into consideration.

***Note:** Additional explanation regarding the functionality of the split tags*

In the following example, there are two usage data block structures that should be copied from the unstructured area ("rm2mraw") to different structured measurement data channels ("histdata0" and "histdata1") using the myDatanet Connector , so that they can be used in conjunction with the myDatanet interface (reports, visualisations, graphics, etc.). The first byte at the start of the data area of a usage data block is used to specify the type of the usage data block structure. Value 0 specifies that the usage data block should be copied to the structured "histdata0" measurement data channel. If the first byte is 1, the usage data block should be copied to the structured "histdata1" measurement data channel. Each usage data block contains header information (time stamp, CRC, etc.) that is not included in the following graphic in the interest of clarity.



Graphic display of the functionality of the split tags

14.3 Data structure

This section describes how the structured measurement data channels ("histdata0" - "histdata9") and configuration memory blocks ("config0" - "config9") are split into individual data fields for use on the Cloud server.

> ***Important note:*** *All of the data fields must be defined via the myDatanet Connector if the structured measurement data channel or configuration block should be available on the Cloud server .*

An extended example, in which most of the available attributes are used, is provided in chapter "Example" on page 284.

Example for the breakdown of a structured measurement data channel ("histdata0") into individual data fields.

Code	Explanation
`<table>`	Start of the table tag
`  name   =histdata0`	Name of the data table to be described
`  <field>`	Start of a field definition (cho) for a
`    name    =ch0`	u16 type field, Description of the attributes, see below
`    title   =Delay`	
`    units   =sec`	
`    type    =u16`	
`    byteofs =0`	
`    vofs    =10`	
`    min     =10`	
`    max     =2000`	
`    chmode  =3`	
`  </field>`	End of a field definition (cho)
`</table>`	

Example for the breakdown of a configuration memory block ("config0") into individual data fields.

Code	Explanation
`<table>`	Start of the table tag
`  name   =config0`	Name of the configuration table to be described
`  <field>`	Start of a field definition (fieldo) for a
`    name   =field0`	u32 type field, Description of the attributes, see below
`    title  =Record interval`	
`    units  =s`	
`    type   =u32`	
`    byteofs=0`	
`  </field>`	End of a field definition (fieldo)
`</table>`	

14.3.1 Attributes of the field definition

name

Alpha-numeric.

- *For structured measurement data channels ("histdata0" - "histdata9"):*

  Internal name of the channel. The rapidM2M has 1000 available channels. The titles ch0 to ch999 can therefore be used.

- *For configuration memory blocks ("config0" - "config9"):*

  Internal name of the configuration parameter. The rapidM2M has 1000 available configuration parameters. The titles field0 to field999 can therefore be used.

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.

type

Data type of the field. The data type simultaneously determines the number of bytes that must be retrieved from the data stream. The only exception is the string (or char) type, for which the length is determined by the max. attribute.
Available data types: s8, s16, s32, s64, u8, u16, u32, f32, f64, string
Alternative titles (standardised values in brackets): bint(s8), wint(s16), dint(s32), qint(s64), byte(u8), word(u16), dword(u32), float32(f32), float(f32), float64(f64), char(string)

param1

Alpha-numeric. Currently the only permissible value: nocodepage
This attribute must be used for fields that contain an ANSI coded string.

byteofs

Numeral, integer positive. Offset of the first byte used for the field, 0-based.

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

Formatting instructions for displaying the field content on the interface of the Cloud server or input via the interface of the Cloud server.

Formatting instructions	*Explanation*
0=off;1=on	*A ComboBox is created in which the text following the "=" is displayed for each entry instead of the value. Entries must be separated using a ";".*

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:

Mode	*Name*	*Explanation*
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 counter*	*A counter 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 counter*	*A counter 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 Cloud 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 Cloud 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.

14.4 Example

`<table>`	Start of the table tag
`  name   =histdata0`	Name of the described data table
`  <field>`	Start of a field definition (ch0)
`    name    =ch0`	Internal name of the channel
`    title   =Delay`	Title of the channel
`    units   =sec`	Units
`    type    =u16`	Data type, u16 means that 2 bytes are used
`    byteofs =0`	The first two bytes of the data stream are used.
`    vofs    =10`	Ten is added to the extracted value.
`    min     =10`	The minimum value for any further use is ten.
`    max     =2000`	The maximum value for any further use is 2000.
`    chmode  =3`	It is an interval counter.
`    index   =1`	Sorting number
`  </field>`	End of the field tag
`  <field>`	
`    name    =ch1`	Internal name of the channel
`    title   =Height`	Title of the channel
`    units   =cm`	Units
`    type    =f32`	Data type, 32 bit floating point
`    byteofs =2`	Bytes 2 to 5 are used.
`    decpl   =2`	Two decimal places
`    vofs    =0`	No offset
`    vscale  =0.01`	The extracted value is multiplied by 0.01.
`    min     =0`	The minimum value for any further use is zero.
`    max     =2000`	The maximum value for any further use is 2000.
`    chmode  =6`	It is an analogue channel.
`    index   =0`	Sorting number
`  </field>`	
`  <field>`	
`    name    =ch2`	Internal name of the channel
`    title   =Pump`	Title of the channel
`    type    =u8`	Data type, 8 bit whole number
`    byteofs =6`	Byte 6 is used.
`    bitmask =01`	The least significant bit (LSB) is masked out of the byte.
`    min     =0`	The minimum value for any further use is zero.
`    max     =1`	The maximum value for any further use is 1.
`    chmode  =1`	It is a digital channel.
`  </field>`	
`  <field>`	
`    name    =ch3`	Internal name of the channel
`    title   =Info`	Title of the channel
`    type    =string`	Data type, string, length is specified in the max. field.
`    byteofs =10`	The text starts at byte 10.
`    max     =50`	The text can have a max. of 50 characters.
`    index   =10`	Sorting number
`  </field>`	
`</table>`	

The specification of the index is missing in ch2, the channel is therefore automatically assigned index 2. The channels are therefore sorted as ch1, ch0, ch2, ch3.

14.5 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 myDatanet is applied.

Overview of the possible values (unsigned):

Value/type	u8 (byte)	u16 (word)	u32 (dword)
NaN	oxFF	oxFFFF	oxFFFFFFFF
OF	oxFE	oxFFFE	oxFFFFFFFE
UF	oxFD	oxFFFD	oxFFFFFFFD
OL	oxFC	oxFFFC	oxFFFFFFFC
SC	oxFB	oxFFFB	oxFFFFFFFB

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	0x7FFFFFFFFFFFFFFE
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	0x7F8xxxxx	0x7FFxxxxxxxxxxxx
OF	Not available	Not available
UF	Not available	Not available
OL	Not available	Not available
SC	Not available	Not available

Only the first 12 bytes are taken into consideration when interpreting NaN for f32 and f64.

6872
6872
6872

Chapter 15 Troubleshooting and repair

15.1 General problems

Problem	Cause/solution
Module does not respond (status LED is always off after the button to switch on the wireless GSM controller is activated)	• Check the cable connections (see "Connecting the sensors, actuators and power supply" on page 93) • Check whether the jumper used to activate the energy supply of the module (J104) is set correctly (see "Jumper to activate the energy supply of the M120 module (J104) " on page 80). • Check that the rapidM2M module M120 is correctly inserted into the holding fixture on the rapidM2M board for developers (see "Inserting the M120 module" on page 92).
Communication problems	• Load the device log from the M120 module using the rapidM2M Toolset (see ""Log" tab" on page 149). A list of all the possible error codes is included in the chapter "Log entries and error codes" (see "Log entries and error codes " on page 289).
Not all or no data is available on the server.	• The connection was interrupted during the transmission, which is indicated by a time-out entry in the connection list (see tbd). Solution: Initiate a transmission or wait for the next cyclical transfer. • The myDatanet Connector was not configured correctly (see "myDatanet Connector " on page 277).
Alarm message was not sent although the alarm is signalled.	• Check the settings of the alarm schedule (see tbd). • Check the address data of the alarm schedule (see tbd).
The pawn script is not being executed correctly.	• Check that the correct script type was selected during the configuration of the control (see "Control" on page 189). • Load the device log from the M120 module using the rapidM2M Toolset (see ""Log" tab" on page 149). A list of all the possible pawn script error codes is included in the chapter "Pawn script error codes" (see "Pawn script error codes" on page 251). • The previous script has been replaced with that of the newly assigned site/application due to a context change (assignment of a different site/application). • By assigning a new or different site/application, the script installed via the USB has been replaced with that of the newly assigned site/application.

Problem	Cause/solution
The buzzer emits a continuous tone.	• Check whether you have set the GPIO1 to "low" via the script. • Check the settings of switch 1 and 2 of dip switch (S207) (see "Output of digital signals" on page 72). The signal output is always stopped by setting both switches to "OFF".
The desired level is not issued at one of the GIPOs.	• Check whether the GPIO has been configured as an output via the "rM2M_GpioDir()" PAWN script function. • Check whether the pull-up or pull-down resistor is activated via dip switch S206 (see "Output of digital signals" on page 72). The output stage of the GPIO is not able to drive the level against the resistor. This means that if pull-up is activated, the output stage cannot set the level to "low".

15.2 Log entries and error codes

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1000	POWER ON	0	---	System start completed
		> 0	---	Internal system error Restart due to internal system error. There may be a hardware problem if the "POWER ON" log entry with a parameter code of >0 is contained in the device log several times. Contact the manufacturer in this case (see "Contact information" on page 307).
1020	ERROR SOD	##	---	Internal system error There may be a hardware problem if this error is contained in the device log several times with the same parameter code. Contact the manufacturer in this case (see "Contact information" on page 307).
1033	MODEM_UPDATE	##	---	Update of the firmware for the modem controller 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).
1161	LOG REFORMATFILE	---	---	The file system was re-formatted. This means that all of the recorded data and log entries were deleted.
1200	MODEM ERROR			Modem error (see "Modem error" on page 292)
1202	MODEM CMME ERROR	##	---	The GPRS modem indicates a +CME error. The parameter specifies the type of error.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1203	SELECTED NETWORK	##	---	A new GSM network was selected. This entry is always duplicated in the device log. In the first entry, the parameter specifies the MCC (Mobile Country Code), while in the second entry it specifies the MNC (Mobile Network Code) of the selected GSM network.
1204	BANNED NETWORK	##	---	A GSM network was added to the ban list. This entry is always duplicated in the device log. In the first entry, the parameter specifies the MCC (Mobile Country Code), while in the second entry it specifies the MNC (Mobile Network Code) of the selected GSM network.
1205	BAN LIST CLEARED	---	---	The ban list was deleted as there were no GSM networks in the receiving range that have not been placed on the ban list.
1212	ERROR MODEM IRREGULAR OFF	##	---	Indicates a faulty connection. The parameter includes a counter that indicates how many consecutive connections have not worked.
1224	MODEM SIM RESET	---	---	The GPRS modem has been restarted to reinitialise the SIM chip again.
1252	MODEM TO CON	##	---	Timeout while a connection is being established. The parameter contains a counter that indicates how many consecutive connection attempts have not worked.
1281	ZLIB STREAMPROCESS ERR	##	---	Internal error Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 307).
1282	ZLIB STREAMFINISH ERR	##	---	Internal error Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 307).

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
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)
2000 - 2999	MODULE ERR			Bereich für die kundenspezifischen Fehlercodes, die mittels der Pawn Script Funktion "rM2M_WriteLog()" in das Gerätelog geschrieben werden können (siehe "rapidM2M functions" on page 198).
3000 - 3099				Error codes of the script execution (see "Pawn script error codes" on page 251).

15.2.1 Modem error

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
GPRS error (1/2)				
1200	BEARER LINE BUSY	-993	---	Line busy Try again later
	BEARER NO ANSWER	-992	---	No answer • Try again later • Check whether the device is in the coverage area (www.microtronics.at/footprint).
	BEARER NO CARRIER	-991	---	No carrier • Try again later • Check whether the device is in the coverage area (www.microtronics.at/footprint).
	BEARER GPRS FAILED	-988	---	GPRS setup failure • Try to improve the position of the antenna • Check whether the device is in the coverage area (www.microtronics.at/footprint).
	BEARER PPP LCP FAILED	-987	---	Error when negotiating the LCP Internal error
	BEARER PPP AUTH FAILED	-986	---	Error occurred during PPP authentication Internal error
	BEARER PPP IPCP FAILED	-985	---	Error when negotiating the IPCP Internal error
	BEARER PPP LINK FAILED	-984	---	PPP receiver does not react to echo request Internal error
	BEARER PPP TERM REQ	-983	---	PPP session terminated by the receiver. Internal error

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
GPRS error (2/2)				
	BEARER CALL REFUSED	-982	---	Incoming call refused Try again later
	BEARER UNKNOWN	-981	---	Try again later
Error when establishing the GPRS connection				
1200	BEARER BAD HDL	-980	---	Invalid handling Internal error
	BEARER BAD STATE	-979	---	The carrier was not stopped. Internal error
	BEARER DEV	-978	---	Error occurred during link layer initialisation Internal error
	BEARER GPRS START UNKNOWN	-977	---	Internal error
Error when disconnecting the GPRS connection				
1200	BEARER GPRS STOP BAD HDL	-976	---	Invalid handling Internal error
	BEARER GPRS STOP UNKNOWN	-975	---	Internal error
1200	NETLOCK FAILURE	-970	---	Error when selecting the network • Check whether the device is in the coverage area (www.microtronics.at/footprint).

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1200	BAND SEL FAILED	-969	---	A network could not be found on the GSM900/1800 or on the GSM85071900 band. • Try to improve the position of the antenna. • Check whether the device is in the coverage area (www.microtronics.at/footprint).
TCP channel error (1/2)				
1200	CHANNEL ABORTED	-965	---	An attempt is being made to write to/read a TCP client that is no longer available. Try again later
	CHANNEL CSTATE	-964	---	The channel is not in the "WIP_CSTATE_ READY" state. Try again later
	CHANNEL NOT SUPPORTED	-963	---	The option is not supported by the channel. Try again later
	CHANNEL OUT OF RANGE	-962	---	The optional value is outside the permissible range. Try again later
	CHANNEL MEMORY	-961	---	Internal error
	CHANNEL INTERNAL	-960	---	Internal error
	CHANNEL INVALID	-959	---	Invalid option or parameter Internal error
	TCP DNS FAILURE	-958	---	The name could not be resolved in an IP address. Internal error
	CHANNEL RESOURCES	-957	---	There are no further TCP buffers available. Try again later
	CHANNEL PORT IN USE	-956	---	TCP server port is already being used. Try again later

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
TCP channel error (2/2)				
1200	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
	CHANNEL UNKNOWN	-950	---	Internal error

6872
6872
6872

Chapter 16 Spare parts and accessories

16.1 rapid M2M module

Description	Quantity	Order number
rapidM2M module M11	1	300068
rapidM2M module M120	1	300087

16.2 Antennas

Description	Quantity	Order number
Magnetic antenna SMA-M 2,5m	1	300085
Extension cable for antenna SMA-M/SMA-F 2,5m	1	206.807

16.3 Power supply

Description	Quantity	Order number
Power supply 24V 1A (U: 24V, I max.: 1A, P max.: 24W)	1	213.814
Power supply 12V 500mA (U: 12V, I max.: 0,5A, P max.: 6W)	1	206.623
Snap-on primary plug EU for chargers/power supplies [1)]	1	300027
Snap-on primary plug UK for chargers/power suppliest [1)]	1	300028
Snap-on primary plug US for chargers/power supplies [1)]	1	300029
Snap-on primary plug AUS for chargers/power supplies [1)]	1	300030

[1)] Not suitable for the plug-in power supply included in the scope of supply of the developer package (300144).

16.4 Adapter

Description	Quantity	Order number
Null modem adapter 9-pin D-Sub female/male	1	206.686

16.5 Other accessories

Description	Quantity	Order number
myDatanet Tool Pen	1	206.646

6872
6872
6872

Chapter 17 Document history

Rev.	Date	Changes
01	22.02.2013	First version
02	10.06.2014	Hardware version 1.2 *The jumper (J113) for selecting the USB supply for the module has been added. This means that "M1x" as well as "M2x" modules can be used.* *The terminal strip for the analogue differential inputs and constant current outputs has been extended by two connectors for the analogue ground.* *The buzzer (LS200) does not buzz anymore when no script is installed in the rapidM2M module M120.* *The "Reset" signal is now connected to pin 13 of the plug connector with the even pin numbers of the M120 module (J107 and J110).* *The jumper (J37) to switch on the wireless GSM controller on the module has been added. (pressing button S101 is not necessary when jumper J37 is set.)* Chapter "Technical data" on page 15 *Specification of the memory size for the PAWN binary has been added* Chapter "Overview" on page 67 *Description adapted to HW version 1.2* Chapter "Jumper for selecting the USB supply for the module (J113)" on page 75 *Chapter added* Chapter "Plug connector to which the even-numbered pins of the M120 module are connected (J107 and J110)" on page 77 *The "Reset" signal is now connected to pin 13 of the plug connector with the even pin numbers of the M120 module (J107 and J110).* Chapter "Jumper to switch on the wireless GSM controller on the rapidM2M module M120 (J37)" on page 80 *Chapter added* Chapter "Technical data" on page 97 *Specification of the memory size for the PAWN binary has been added* Chapter "52pin connector strip (X2)" on page 100 *Detail overview of the connector strip added* Chapter "Memory organisation" on page 103 *Chapter added.* Chapter "Technical details about the energy supply" on page 105 *Table with information regarding the required supply voltages and current consumptions added*

Rev.	Date	Changes
02	10.06.2014	**Chapter "ENPower signal" on page 106** *Chapter added* **Chapter "Technical details about the level shifters" on page 107** *Information on the current consumption of the level shifters added* **Chapter "Typical application" on page 112** *Chapter added* **Chapter "Cloud server" on page 115** *Screenshots and explanations adapted to server version V39.2* **Chapter "New registration and logging in" on page 115** *Instructions adapted to the current version of the homepage* **Chapter ""Statistics" area" on page 134** *Chapter added.* **Chapter "rapidM2M Toolset " on page 137** *Screenshots and explanations adapted to version 1.6* **Chapter "Generally applicable table identifiers" on page 156** *Chapter added* **Chapter "Device-specific table identifier" on page 162** *Chapter added.* **Chapter "Control" on page 189** *Description on uploading binary files to the server has been added* *Script templates are no longer supported by server version V39.2.* **Chapter "Configuration 0 - configuration 1" on page 190** *Chapter added* **Chapter "Pawn script" on page 193** *Structure of the sub-chapters revised* **Chapter "General" on page 193** *Description on uploading binary files to the server has been added* *Script templates are no longer supported by server version V39.2.* **Chapter "rapidM2M API" on page 194** *Specification of the include file is now only required for string functions.* **Chapter "Configuration of a HTTP-GET command" on page 166** *Explanation of the optional "verbose_time" parameter added* **Chapter "Data record compression" on page 172** *More detailed description of the summation method added.*

Rev.	Date	Changes
03	05.03.2015	**Chapter "Technical data" on page 15** *Specification of the sizes of the memory areas (data memory, configuration memory and registration memory) updated in accordance with firmware version 00v014* **Chapter "General product information" on page 19** *Information regarding the registration memory blocks added* **Chapter "Tutorial" on page 23** *Chapter added* **Chapter "Recommended procedure" on page 58** *Chapter added* **Chapter "Procedure in case of connection aborts" on page 60** *Chapter added* **Chapter "Error handling" on page 62** *Chapter added* **Chapter "Registration memory blocks" on page 62** *Chapter added* **Chapter "File transfer" on page 64** *Chapter added* **Chapter "Meaning of the SIM state" on page 64** *Chapter added* **Chapter "RS232 interfaces" on page 71** *Note added indicating that the RS232 level converters on the developer board must be removed if the UART signals are supplied directly via the plug connector to which the pins with the uneven numbers of the M120 module are connected.* *The hardware handshake is now supported by the firmware.* **Chapter "Simulation circuits" on page 72** *Correction of the previous version: The buzzer emits a sound when the GPIO1 is set to "low".* **Chapter "Plug connector to which the even-numbered pins of the M120 module are connected (J107 and J110)" on page 77** *Correction of the previous version: The GPIO6 is connected to pin 26 of the plug connector.* **Chapter "Plug connector to which the odd-numbered pins of the M120 module are connected (J106 and J109)" on page 75** *Note added indicating that the RS232 level converters on the developer board must be removed if the UART signals are supplied directly via this plug connector.* **Chapter "Supply (J105 or J108) " on page 79** *Information about the reverse voltage protection for the supply voltage input has been added.*

Rev.	Date	Changes
03	05.03.2015	**Chapter "Technical data" on page 97** *Specification of the sizes of the memory areas (data memory, configuration memory and registration memory) updated in accordance with firmware version 00v014* **Chapter "Memory organisation" on page 103** *Information regarding the registration memory blocks added* **Chapter "UART" on page 108** *More detailed description of the assignment of the UART of the M120 module and COM connector of the developer board* **Chapter "Interrupt inputs" on page 112** *More detailed explanation regarding the differences between the GPIO and interrupt inputs added.* **Chapter "Typical application" on page 112** *Information about the reverse voltage protection for the supply voltage input has been added.* **Chapter "Sites" on page 121** *Explanation of the application-specific control elements added* **Chapter "Creating a new site" on page 123** *Explanation on selecting an application template added* **Chapter "List of files" on page 125** *Chapter added* **Chapter ""User" area" on page 128** *Explanation of the search field added* **Chapter "Installation" on page 139** *Information added indicating that two USB drivers are now installed* **Chapter "Overview" on page 142** *Information added indicating that a site must be assigned to the module before it is accessed via USB* **Chapter ""Script" tab" on page 144** *Description of the interface adapted to version 1.12* **Chapter ""Server" tab" on page 149** *Explanation added indicating that this tab now forwards you to the test page of the REST-API* **Chapter ""FW" tab" on page 151** *OS version of the module is now also displayed.* **Chapter "Hot keys" on page 152** *Chapter added*

Rev.	Date	Changes
03	05.03.2015	**Chapter "Structure of the database" on page 153** *List of device-specific table identifiers extended to include the measurement data channels ("histdata0" - "histdata9"), device log, device status, localisation information and registration memory blocks* *Detailed explanation added regarding the assignment of the tables to the device or site* **Chapter "rm2mlog" on page 162** *Chapter added* **Chapter "rm2mstate" on page 162** *Chapter added* **Chapter "rm2mlocals" on page 163** *Chapter added* **Chapter "rm2mpos" on page 163** *List of functions that can create the data for this table has been supplemented.* **Chapter "rm2m_ctrlscript" on page 164** *Explanation of "script_error" field added* **Chapter "reg_sys_otp" on page 165** *Chapter added* **Chapter "reg_sys_flash" on page 165** *Chapter added* **Chapter "reg_app_otp" on page 165** *Chapter added* **Chapter "reg_app_flash" on page 166** *Chapter added* **Chapter "Data record compression" on page 172** *Information added indicating that JSON documents cannot be issued when data compression is activated* **Chapter "Output formats of the HTTP-GET command" on page 177** *Chapter added* **Chapter "Transferring data to the Cloud server" on page 179** *Detailed explanation for using the "Wget" command line tool added* **Chapter "Control" on page 189** *Information added indicating that this configuration section is only visible if the site was not created from an application template* **Chapter "Measurement instrument" on page 191** *Explanation of "Hardware version" field added* **Chapter "Console functions" on page 197** *Explanation of the return values supplemented*

Rev.	Date	Changes
03	05.03.2015	**Chapter "rapidM2M functions" on page 198** *Explanation of the arrays including symbolic indices added* *Explanation of the constants added* *Explanation of the callback function added* *Explanation of the "rM2M_GetId", "rM2M_GetTime" "rM2M_GetDate" functions updated in accordance with firmware version 00v014* *Explanation of the "rM2M_GetTimezoneOffset", "rM2M_Pack", "rM2M_CfgInit", "rM2M_CfgWrite", "rM2M_CfgFlush", "rM2M_RegGetString", "rM2M_RegSetString", "rM2M_RegGetValue", "rM2M_RegSetValue", "rM2M_RegOnChg", "rM2M_SmsInit", "rM2M_SmsClose", "rM2M_WriteLog", "rM2M_SetPos", "rM2M_SetPosNMEA", "rM2M_GetPos", "rM2M_GetGSMPos" and "rM2M_GSMGetInfo" functions added* *Explanation of the return values supplemented* **Chapter "File transfer functions" on page 236** *Chapter added* **Chapter "Help functions" on page 248** *Explanation of the arrays including symbolic indices added* *Explanation of the return values supplemented* *Explanation of the "CalcTable" and "CalcTableF" functions added* **Chapter "Pawn script error codes" on page 251** *Explanation of "SCRIPT_ERR" error code added* *Possible causes for the "AMX_ERR_EXIT" and "AMX_ERR_MEMACCESS" error codes added* **Chapter "Split tag" on page 278** *Explanation of the configuration memory blocks revised.* **Chapter "myDatanet Connector " on page 277** *Explanation of the configuration memory blocks revised.* **Chapter "Data structure" on page 280** *Explanation of the "name" attributes relating to the configuration memory blocks revised.* *Explanation of the "para1", "editmask", "view" and "edit" attributes added* **Chapter "Troubleshooting and repair" on page 287** *Possible troubleshooting in the event that the pawn script is not executed correctly, the buzzer emits a continuous tone or the desired level is not issued at the GPIOs* **Chapter "Log entries and error codes " on page 289** *Explanation of the "POWER ON", "LOG REFORMATFILE", "MODEM SIM RESET" and "SIM_STATE" log entries added* **Chapter "Glossary" on page 305** *Chapter added*

Chapter 18 Glossary

Footprint

The manufacturer's devices are equipped with subscriber identity modules (SIM) ex-works for the purpose of mobile data transmission. The footprint describes the countries and regions, in which a mobile connection is available (see www.microtronics.at/footprint).

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.

6872
6872
6872

Chapter 19 Contact information

Support & Service:
Microtronics Engineering GmbH
Hauptstrasse 7
3244 Ruprechtshofen
Austria, Europe
Tel. +43 (0)2756 7718023
support@microtronics.at
www.microtronics.at

Microtronics Engineering GmbH (Headquarter)
Hauptstrasse 7
3244 Ruprechtshofen
Austria, Europe
Tel. +43 (0)2756 77180
Fax. +43 (0)2756 7718033
office@microtronics.at
www.microtronics.at

Microtronics Engineering GmbH
Windthorststrasse 5
63179 Obertshausen
Germany, Europe
office@microtronics.de
www.microtronics.de