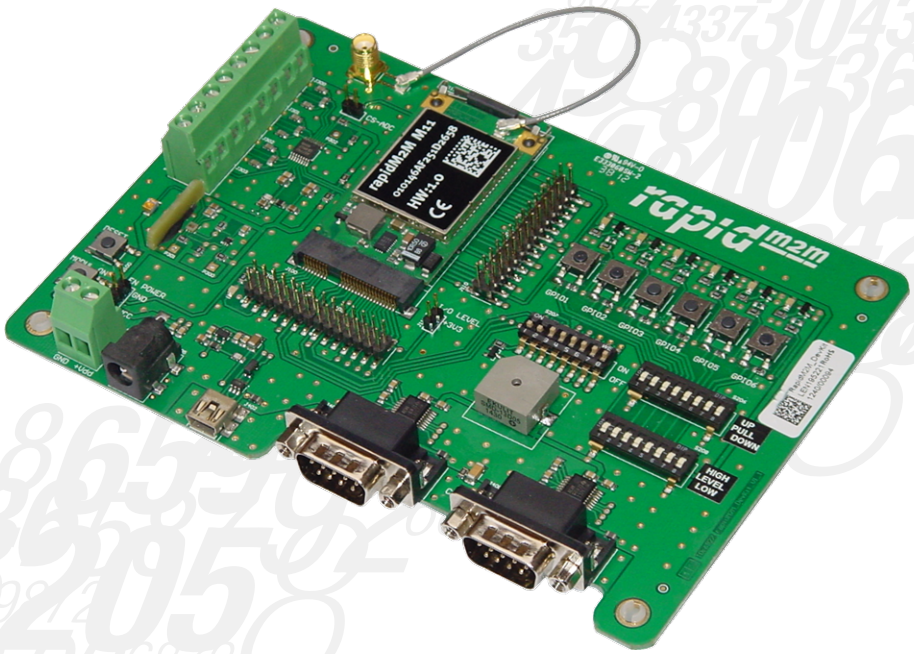
User manual for the rapidM2M developer package M11



Rev. 02 valid from:

- Firmware version: ---
- Modem version: 03v002
- Hardware version: 1.0

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 M11 1

Chapter 1 Table of contents 3

Chapter 2 Technical data 11

Chapter 3 General specifications 13

3.1 General product information 13

3.2 Scope of supply 13

3.3 Dimensions 14

3.4 Device labelling 14

3.5 Storage of the product 15

Chapter 4 Functional Principle 17

4.1 ALOHA transmission mode 17

4.2 Selection of the GSM network 17

4.2.1 Ban list 18

4.3 Script eradication 19

4.3.1 Instruction list 19

4.3.2 Pawn 19

Chapter 5 rapidM2M Developer Board 21

5.1 Overview 21

5.1.1 Block diagram 23

5.1.2 RS232 interface COM1 (J400) 23

5.1.3 Jumper for supplying the level shifters (J111) 24

5.1.4 RS232 interface COM2 (J401) 24

5.1.5 Plug connector to which the odd-numbered pins of the M11 module are connected (J106 and J109) 25

5.1.6 Status LED (LED100) 27

5.1.7 Supply (J105 or J108) 27

5.1.7.1 Terminal strip (J108) 27

5.1.7.2 DC coaxial connector (J105) 28

5.1.8 Jumper to activate the energy supply of the M11 module (J104) ... 28
5.1.9 Button to switch on the wireless GSM controller on the rapidM2M module M11 (S101) ... 28
5.1.10 Button to reset the wireless GSM controller on the rapidM2M module M11 (S100) ... 29
5.1.11 Holding fixture for the M11 module ... 29
5.1.12 Plug connector to which the even-numbered pins of the M11 module are connected (J107 and J110) ... 29
5.2 Circuit diagram ... 31
5.2.1 Overview ... 31
5.2.2 Connector ... 32
5.2.3 USB RS232 ... 33
5.2.4 Sensors ... 34
5.2.5 GPIO ... 35
5.3 Layout ... 36
5.4 Electrical installation ... 37
5.4.1 Information on preventing electrostatic discharges (ESD) ... 37
5.4.2 Inserting the M11 module ... 38
5.4.3 Connection of the sensor and power supply ... 39
5.4.4 Connecting the GSM antenna ... 40
Chapter 6 rapidM2M module M11 ... 43
6.1 Technical data ... 43
6.2 Overview ... 44
6.2.1 Block diagram ... 45
6.2.2 52pin connector strip (X2) ... 46
6.2.3 Device labelling ... 47
6.3 Dimensions ... 47
6.3.1 Dimensions of the 52pin connector strip (X2) ... 48
6.4 Technical details ... 48
6.4.1 Technical details regarding the energy supply ... 48
6.4.1.1 Supply via VIN ... 48

6.4.1.2 Supply via VAUX ... 48
6.4.2 ENCPU signal ... 48
6.4.3 Reset signal ... 49
6.4.4 Status LED ... 50
6.4.5 Technical details about the level shifters ... 50
6.4.6 Technical details regarding UART1 ... 50
6.4.7 Technical details regarding UART2 ... 51
6.4.7.1 Error codes of the UART2 interface ... 51
Chapter 7 Placing into operation ... 53
7.1 Placing the system into operation ... 53
7.2 Testing communication with the device ... 53
Chapter 8 User interface on the myDataweb-Server ... 55
8.1 Site configuration ... 55
8.1.1 Site ... 55
8.1.2 Comments ... 55
8.1.3 Control ... 55
8.1.4 Interface channels 1-32 ... 56
8.1.4.1 Basis ... 56
8.1.4.2 Alarms ... 56
8.1.4.3 Trigger ... 57
8.1.5 Interface channels 33-64 ... 57
8.1.5.1 Basis ... 57
8.1.6 Interface output channels 1-32 ... 57
8.1.7 Interface output channels 33-64 ... 58
8.1.8 Calculated channels ... 58
8.1.8.1 Basis ... 58
8.1.8.2 Calculation ... 59
8.1.8.3 Alarm ... 59
8.1.9 Internal channels ... 59
8.1.9.1 Basis ... 59

8.1.9.2 Alarms ... 59
8.1.9.3 Trigger ... 60
8.1.10 Serial interface ... 60
8.1.11 Alarm settings ... 65
8.1.12 Basic settings ... 66
8.1.13 FTP export settings ... 67
8.2 Device configuration ... 67
8.2.1 Comments ... 68
8.2.2 Measurement instrument ... 68
8.2.3 Device-specific settings ... 69
8.2.4 GPRS ... 69
Chapter 9 Instruction list ... 71
9.1 General ... 71
9.1.1 Direct entry of an instruction list ... 71
9.1.2 Using a script template ... 71
9.2 Implementation details ... 72
9.2.1 Common elements ... 72
9.2.2 IL syntax elements ... 74
9.3 Instruction list error codes ... 76
9.4 Expansions ... 78
9.5 Device-specific implementation ... 80
9.6 Example ... 81
9.6.1 Saw-tooth generator ... 81
9.6.2 Weekday designation ... 82
Chapter 10 Pawn script ... 85
10.1 General ... 85
10.1.1 Direct input of a pawn script ... 85
10.1.2 Using a script template ... 85
10.2 General syntax ... 85
10.2.1 Format ... 85

10.2.2 Optional semicolons ... 85
10.2.3 Comments ... 86
10.2.4 Identifier ... 86
10.2.5 Reserved keywords ... 86
10.2.6 Numerical constants ... 86
10.2.6.1 Numerical integer constants ... 86
10.2.6.2 Numerical floating-point constants ... 87
10.3 Variables ... 87
10.3.1 Declaration ... 87
10.3.2 Local declaration ... 87
10.3.3 Global declaration ... 87
10.3.4 Static local declaration ... 88
10.3.5 Static global declaration ... 88
10.3.6 Floating point values ... 88
10.4 Constant variables ... 88
10.5 Array variables ... 89
10.5.1 One-dimensional arrays ... 89
10.5.2 Initialisation ... 89
10.5.3 Progressive initialisation for arrays ... 89
10.5.4 Multi-dimensional arrays ... 90
10.5.5 Arrays and the "sizeof" operator ... 90
10.6 Operators and expressions ... 91
10.6.1 Notational conventions ... 91
10.6.2 Expressions ... 91
10.6.3 Arithmetic ... 92
10.6.4 Bit manipulation ... 92
10.6.5 Assignment ... 92
10.6.6 Comparative operators ... 93
10.6.7 Boolean ... 94
10.6.8 Other ... 95

10.6.9 Priority of the operators ... 95
10.7 Statements ... 96
10.7.1 Statement label ... 97
10.7.2 Composite statements ... 97
10.7.3 Expression statement ... 97
10.7.4 Empty statement ... 97
10.7.5 Assert expression ... 97
10.7.6 Break ... 98
10.7.7 Continue ... 98
10.7.8 Do statement while (expression) ... 99
10.7.9 Exit expression ... 99
10.7.10 For (expression 1; expression 2; expression 3) statement ... 99
10.7.11 Goto label ... 100
10.7.12 If (expression) statement 1 else statement 2 ... 100
10.7.13 Return expression ... 101
10.7.14 switch (expression) {case list} ... 101
10.7.15 While (expression) statement ... 102
10.8 Functions ... 103
10.8.1 Function arguments ("call-by-value" versus "call-by-reference") ... 104
10.8.2 Named parameters versus fixed parameters ... 105
10.8.3 Standard values of function arguments ... 106
10.9 Extensions ... 108
10.9.1 Constants ... 108
10.9.2 Device-dependent channel numbers ... 109
10.9.3 Functions ... 110
10.9.3.1 Core functions ... 110
10.9.3.2 Console functions ... 113
10.9.3.3 Mdn functions ... 114
10.9.3.4 Mathematical functions ... 117
10.9.4 Pawn script error codes ... 119

10.10 Example ... 120
10.10.1 Saw-tooth generator ... 120
10.10.2 Weekday designation ... 121
10.10.3 Calculating the flow rate with the table ... 123
10.11 Differences to C ... 125
Chapter 11 Troubleshooting ... 127
11.1 General problems ... 127
11.2 Log entries and error codes ... 130
11.2.1 Modem error ... 133
11.3 Evaluating the device log ... 136
11.3.1 Evaluating the device log on the myDataweb server ... 136
11.3.2 Evaluating the device log using myDatanetDeviceConfig ... 136
Chapter 12 Spare parts and accessories ... 137
12.1 rapid M2M module ... 137
12.2 Antennas ... 137
12.3 Power supply ... 137
12.4 Adapter ... 137
12.5 Other accessories ... 137
Chapter 13 Application example ... 139
13.1 Lantern control ... 139
13.1.1 Specifications ... 139
13.1.2 Required configuration ... 140
13.1.3 Instruction list ... 141
13.1.4 Communication on the interface ... 142
Chapter 14 Document history ... 143
Chapter 15 Contact information ... 145

6872
6872
6872

Chapter 2 Technical data

Note: *The following technical specifications apply to the rapidM2M Developer Board with the rapidM2M module M11 installed. A detailed technical description of the rapidM2M module M11 is provided in chapter "Technical data" on page 43.*

Voltage supply	5...32 VDC , max. t.b.d. mA@12V
Housing	Weight: 134g (incl. rapidM2M module M11) 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
Display	LED for signalling the operating state
Antenna connector	SMA
Interfaces	1x RS232 for the connection of a digital sensor • Baud rate: 300-115200 • Stop bits: 1,2 • Parity:N, E, O • Data bits: 7, 8 Modes: • ASCII Number of processable interface channels • Inputs: 64 • Outputs: 64 1x RS232 for the output of debug information Additional information is provided in "RS232 interface COM1 (J400)" on page 23 and "RS232 interface COM1 (J400)" on page 23.
Data memory	Internal flash memory for up to 2620 measurement cycles
Data type	f32 (32 Bit floating point)
Data transmission	By means of GSM/GPRS quad-band modem to the respective myDataweb server
SIM	The rapidM2M module M11 is equipped with an integrated SIM chip.
Monthly data volume	t.b.d. at two-minute measurement cycle and 120-minute transmission cycle

6872
6872
6872

Chapter 3 General specifications

3.1 General product information

It is a developer package that is designed to help the customer to integrate the rapidM2M module M11 in his own devices. The rapidM2M Developer Board is therefore fitted with standardised connectors, plug connectors, terminal strips, buttons and LEDs (see "Overview" on page 21) to ensure test connections can be established easily and efficiently. The rapidM2M module M11 included in the scope of supply of the rapidM2M developer package M11 is a module for recording and transmitting measurement data taken from the ASCII data message of a digital sensor. 64 interface channels are provided for this purpose, whereby the physical connection between the rapidM2M Developer Board and the digital sensor is established via a RS232 interface.

> ***Note:*** *Please note for the subsequent integration of the M11 module in your own hardware that the RS232 level converter is integrated in the rapidM2M Developer Board . The output level of the UART of the M11 module is dependent on the power supply of the module's level shifters (see "Technical details about the level shifters" on page 50).*

It is also possible to transmit data back to the digital sensor via the same RS232 interface. 64 output channels are provided for this purpose. The output data can either be specified via the myDataweb input screen or can be calculated by the rapidM2M module M11 after each measurement cycle by means of a script (see "Instruction list" on page 71 or "Pawn script" on page 85). The measurement data from the interface channels is temporarily saved to an internal data memory and wirelessly transmitted to a central location at freely selectable intervals. The M11 module is also configured via this connection. The device is equipped with an integrated SIM chip.

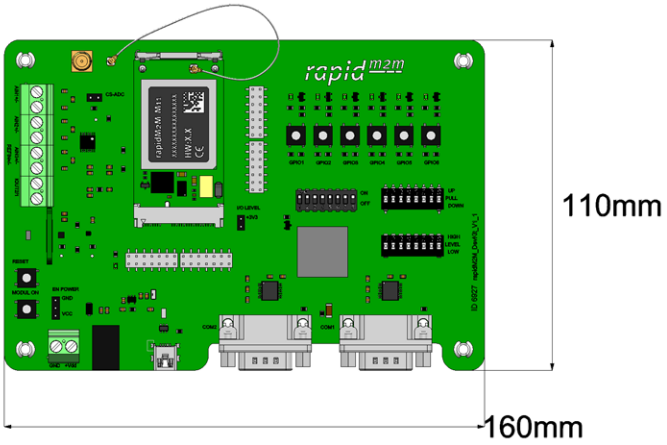
3.2 Scope of supply

The standard scope of supply of the developer package (300070) includes:

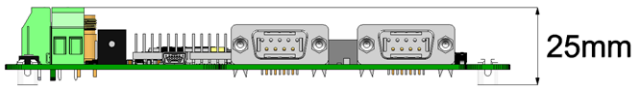
- rapidM2M Developer Board
- rapidM2M module M11 (300068)
- Plug-in power supply (6W 12V)
- Magnetic roof antenna SMA (300085)
- 10 x jumpers
- U.FL antenna cable
- myDatanet Tool Pen (206.646)
- 4 x spacers
- Quick Guide (300084)

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



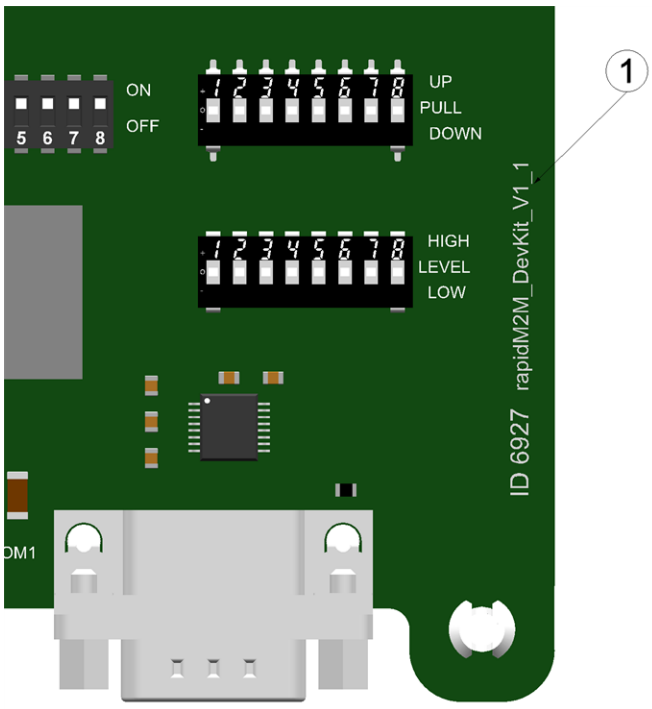
Dimensions: Width and depth



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 transmitted to the myDataweb server. As the rapidM2M module M11 does not support the connection of a button 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 myDataweb server. This particularly applies to the "interval" connection type (see "Basic settings" on page 66). If the "interval and wakeup" connection type has been selected, you can initiate the ALOHA transmission mode via the myDataweb server to transmit the data. Then check again to make sure all of the relevant data has been transmitted. With the "online" connection type, the determined measurement data is immediately transmitted to the myDataweb 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 Developer Board from the supply voltage. If possible, switch off the power supply for the digital sensor before disconnecting the RS232 connection between the rapidM2M Developer Board and the digital sensor. The antenna can then be removed. Store the rapidM2M developer package M11 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 continue to work without an energy 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 myDataweb server.

6872
6872
6872

Chapter 4 Functional Principle

4.1 ALOHA transmission mode

ALOHA transmission mode is a special connection mode whereby the rapidM2M module M11 establishes a connection to the myDataweb server for a period of time configured via the "Basic settings" configuration section (see "Aloha/wakeup duration" in chapter "Basic settings" on page 66).

Initiating ALOHA transmission mode:

- Via the myDataweb server if the "interval and wakeup" connection type has been selected (see "Basic settings" on page 66).



A speech bubble in the measurement device list (see "myDataweb-Server user manual " 206.886) with the title "Aloha" indicates that a device is in ALOHA transmission mode.



Clicking on the speech bubble with the "Aloha" title opens the ALOHA data window (see "myDataweb-Server user manual " 206.886), which comprises the internal "GSM level" measurement value. The measurement values for the last thirty minutes are always shown in the ALOHA data window, this can mean that data from a previous ALOHA transmission is included. The ALOHA data is generated every three seconds independently of the normal measurement values and is thus not saved with the standard measurement data.

If a measurement dataset is recorded during ALOHA transmission mode, it is immediately transferred to the myDataweb server and saved in the standard measurement data.

If a display element that uses one of the device's measurement values is shown on the server interface, this data is also transmitted to the myDataweb server each time a measurement value is generated. If the measurement is completed more frequently than the recording ("measurement cycle" is not equal to zero and/or "burst interval" is not equal to zero) (see "Example for clarification of the recoding, measurement and burst interval" on page 67), this increases the data transmission. However, this data is not permanently stored on the server or on the device, but is instead used to update the display elements on the server (see "myDataweb-Server user manual " 206.886).

4.2 Selection of the GSM network

The GSM network to which the device should register must be selected, as the rapidM2M developer package M11 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 but 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 M11 logs itself into a GSM network:

- PowerOn
- Expiration of the transmission cycle if the "Interval" connection type was selected (see "Basic settings" on page 66).
- Activation of the "Interval & Wakeup" or "Online" connection type (see "Basic settings" on page 66)

1. The rapidM2M developer package M11 checks whether any errors occurred during the last login attempt.

   No errors:

   The rapidM2M developer package M11 logs into the same GSM network that was also used last time.

   Error:

   Continue with the next step

   > ***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 first 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 18.*

4.2.1 Ban list

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

If a "BEARER GPRS FAILED" error occurs (see "Modem error" on page 133), an internal counter is increased by 2 . If the 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 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

coverage area is completed once the ban list is deleted and the network selection is completed with the ban list that is now empty (see item 3 in the chapter"Selection of the GSM network" on page 17).

4.3 Script eradication

4.3.1 Instruction list

The instruction list is run through at the time of every measurement once all of the measurement values have been generated. If an error occurs during this process, the script execution will stop and enter the relevant error code in the device log (see "Instruction list error codes" on page 76). Only then will the measured data and data calculated by the instruction list be recorded, if the record interval has also expired.

> ***Note:*** *If an instruction list is used to write directly to an output, the setpoint entered via the input screen of the myDataweb server is overwritten with the calculated value.*

4.3.2 Pawn

The PAWN script must include the following two functions:

main();
This function is executed during a PowerOn and when the PAWN script is exchanged. It should include all initialisations that only have to executed once during program start-up.

Mdn_CtrlFinsih();
This function is called up at the time of every measurement once all of the measurement values have been generated. It should comprise all of the calculations and functions that should be performed cyclically.

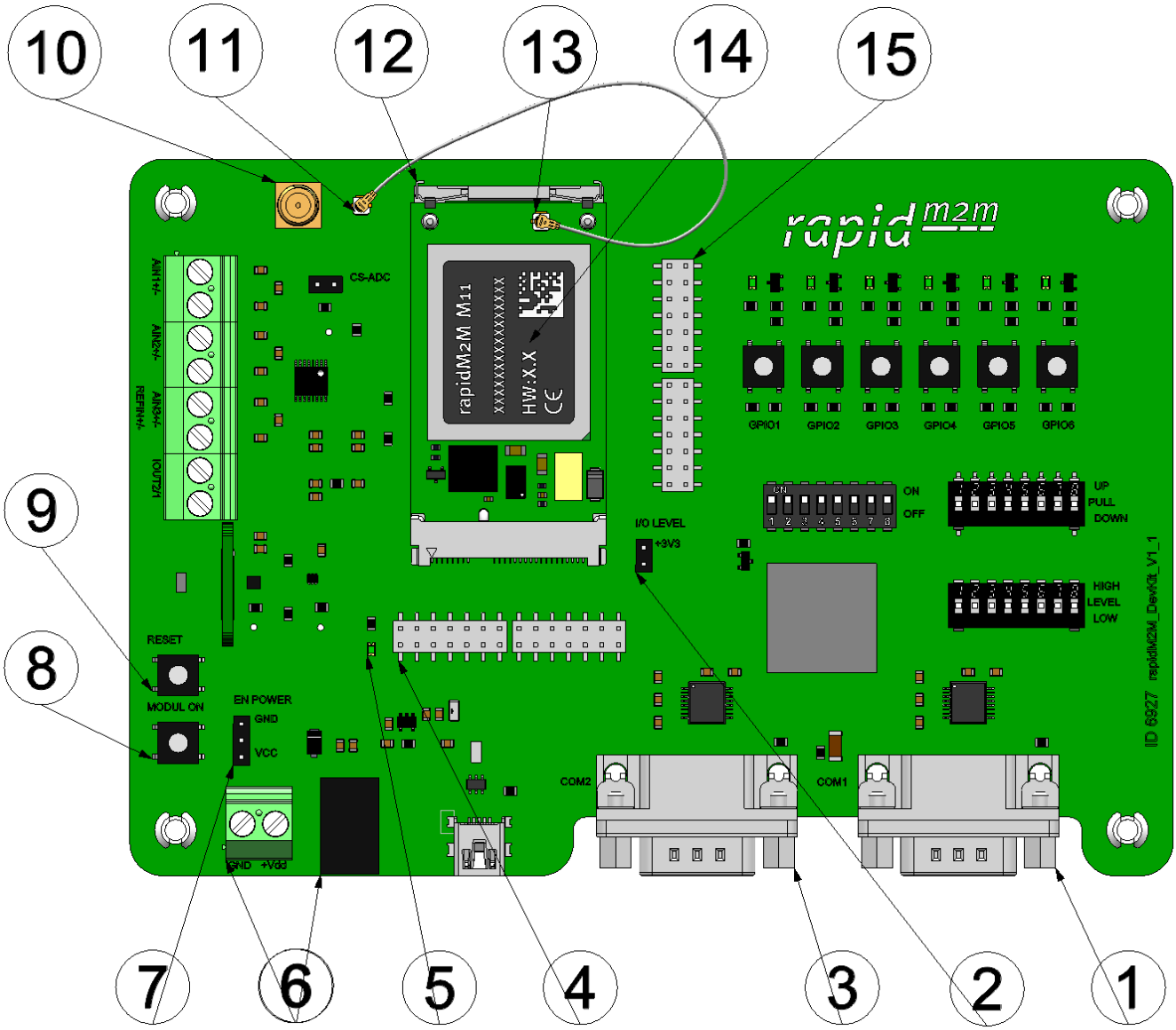
> ***Note:*** *If a pawn script is used to write directly to an output, the setpoint entered via the input screen of the myDataweb server is overwritten with the calculated value.*

If an error occurs during this process, the script execution is stopped and deactivated. The relevant error code is also entered in the device log (see "Pawn script error codes" on page 119).

6872
6872
6872

Chapter 5 rapidM2M Developer Board

5.1 Overview

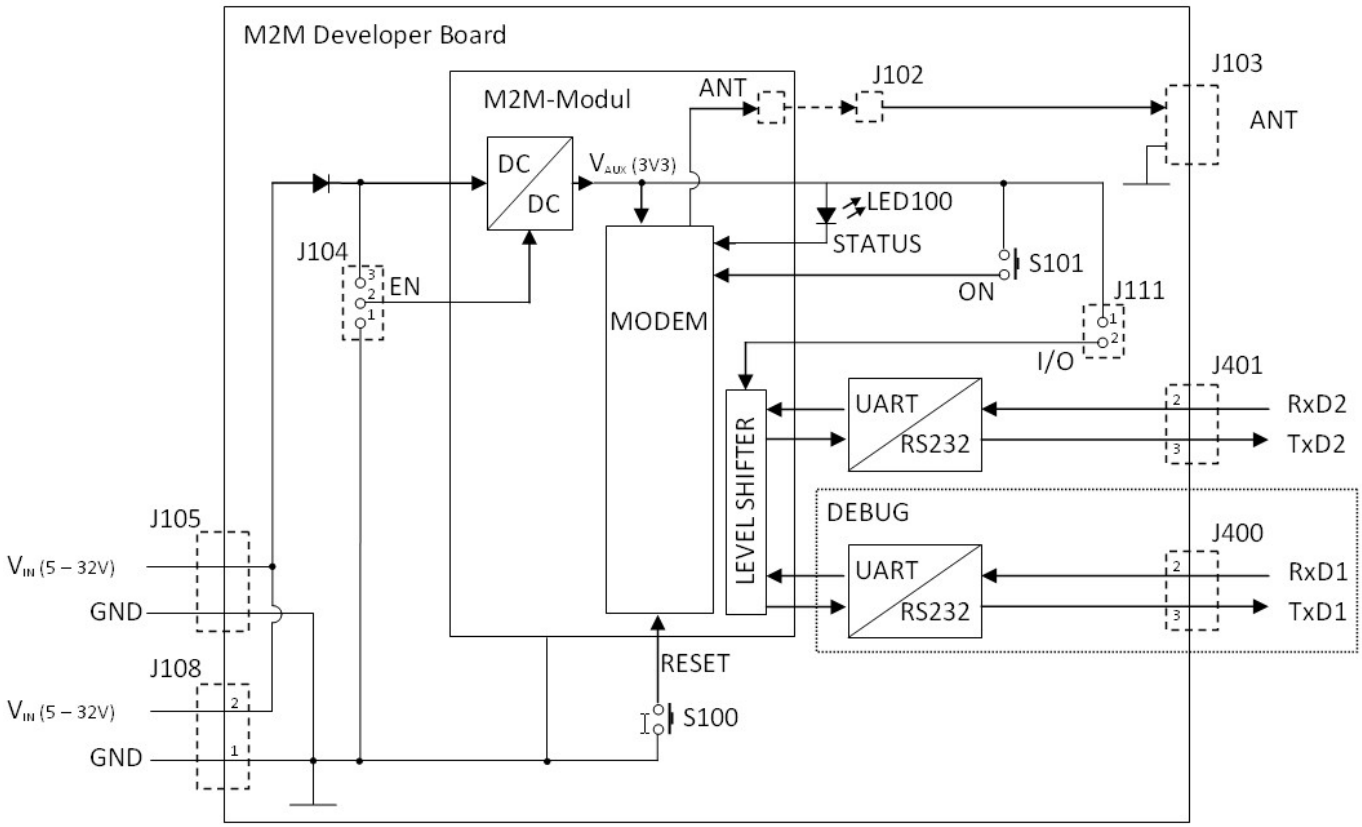


Overview rapidM2M developer package M11

1	COM1 (J400)	9	Button to reset the wireless GSM controller on the rapidM2M module M11 (S100)
2	Jumper for supplying the level shifters (J111)	10	SMA antenna connector (J103) (see "Connecting the GSM antenna" on page 40)
3	COM2 (J401)	11	U.FL antenna connector to connect the M11 module with the rapidM2M Developer Board (J102)
4	Plug connector to which the odd-numbered pins of the M11 module are connected. (J106 and J109)	12	Holding fixture for the M11 module (see "Inserting the M11 module" on page 38)
5	Status LED (LED100)	13	U.FL antenna connector of the M11 module
6	Supply (DC coaxial connector (J105) or terminal strip (J108))	14	rapidM2M module M11 (see "rapidM2M module M11 " on page 43)
7	Jumper to activate the energy supply of the M11 module (J104)	15	Plug connector to which the even-numbered pins of the M11 module are connected. (J107 and J110)
8	Button to switch on the wireless GSM controller on the rapidM2M module M11 (S101)		

5.1.1 Block diagram

Note: A detailed block diagram of the M11 module inserted in the rapidM2M Developer Board is provided in chapter "Block diagram" on page 45.



Block diagram of the developer board

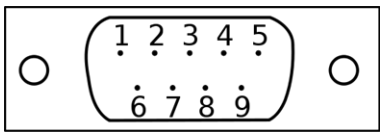
5.1.2 RS232 interface COM1 (J400)

Important note: The UART1 of the M11 module is designed for the output of debug information, whereby the hardware handshake is not active. The UART1 cannot be used to record data.

The UART1 of the M11 module can be accessed via the COM1 connector of the developer board. Conversion to the RS232 level is completed by means of a level converter located on the rapidM2M Developer Board (see "Block diagram" on page 23).

Baud rate	115200
Stop bits	2
Parity	N
Data bits	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 [1)]
8	CTS [1)]
9	NC

[1)] The current firmware of the M11 module does not support the hardware handeshake.

5.1.3 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 M11 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 M11 module is supplied back into the module via jumper J111 on the developer board to supply the level shifters (see "Block diagram" on page 23).

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

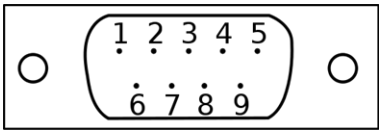
5.1.4 RS232 interface COM2 (J401)

> ***Important note:*** *The rapidM2M module M11 included in the scope of supply of the developer package does not support a hardware handshake.*

The UART2 of the M11 module can be accessed via the COM2 connector of the developer board. Conversion to the RS232 level is completed by means of a level converter located on the rapidM2M Developer Board (see "Block diagram" on page 23).

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).



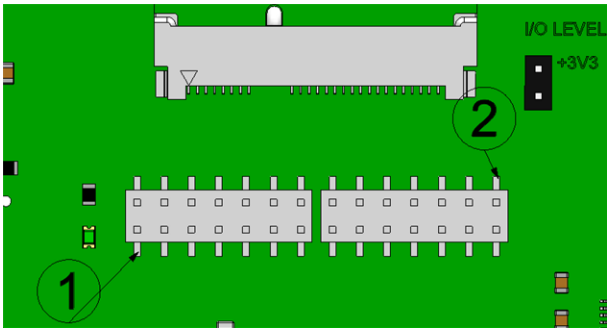
9-pin Sub-D(m)

Assignment of the Sub-D connector

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

[1)] The current firmware of the M11 module does not support the hardware handeshake.

5.1.5 Plug connector to which the odd-numbered pins of the M11 module are connected (J106 and J109)



Plug connector to which the odd-numbered pins of the M11 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 M11
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 M11
5-6		GND	Mass
7-8			Reserved for extensions[1)]
9	I	V_{IN}	Supply voltage (4,5...32VDC) for the rapidM2M module M11
10.15	I/O	V_{AUX}	3,3V voltage output if the rapidM2M module M11 is supplied via V_{IN} Supply voltage input (3,2...4,5V) if no voltage is applied to V_{IN}
11-14			Reserved for extensions[1)]
16	I/O	I/O level	2,8...5,5V supply of the level shifters to adjust the signal level of the M11 module to the surrounding hardware
17-18			Reserved for extensions[1)]
19-20		GND	Mass
21	O	RTS2	RTS line of COM2[2)]
22	I	CTS2	CTS line of COM2[2)]
23	O	TxD2	TxD line of COM2
24	I	RxD2	RxD line of COM2
25	O	RTS1	RTS line of COM1[2)]
26	I	CTS1	CTS line of COM1[2)]
27	O	TxD1	TxD line of COM1
28	I	RxD1	RxD line of COM1

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

[2)] The current firmware of the M11 module does not support the hardware handeshake.

5.1.6 Status LED (LED100)

The status LED is used to indicate the current operating state.

Operating states

Status LED	Description
Off	The rapidM2M module M11 is not connected to the power supply.
Lights up	The rapidM2M module M11 is connected to the power supply but has not yet been logged into the network.
Flashes t_{on}/t_{off}: 0,2/2sec.	The rapidM2M module M11 is logged into the network.
Fast flashing t_{on}/t_{off}: 0,2/0,6sec.	The rapidM2M module M11 is connected to the myDataweb server. (e.g. online mode or ALOHA transmission mode)

5.1.7 Supply (J105 or J108)

The supply of the developer board is either provided via the DC coaxial connector (J105) or the terminal strip (J108). 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.

5.1.7.1 Terminal strip (J108)

U	5...32 VDC
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	Mass
2	$+V_{dd}$	Supply voltage:5...32 VDC

5.1.7.2 DC coaxial connector (J105)

U	5...32 VDC
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

5.1.8 Jumper to activate the energy supply of the M11 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 M11 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 M11 module has reached the maximum value.

5.1.9 Button to switch on the wireless GSM controller on the rapidM2M module M11 (S101)

50ms (soft start) after the energy supply of the M11 module has been activated by means of jumper J104 (see "Jumper to activate the energy supply of the M11 module (J104) " on page 28), the wireless GSM controller of the M11 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 following message is issued on the debug interface COM1 (J401) during the boot process:

```
##########################################################
# myDatalogAIR232       Build Aug 28 2012 13:06:25 03v000 #
##########################################################
# OS        # OK    # 7.46.0
# S/N       # OK    # 010146AF25120BCD
# GSM       # OK    # -69 dBm
# IMSI      # OK    # 232034011697750
# GPRS      # OK    # IP = xxx.xxx.xxx.xxx
# MNT SERVER # OK    # connected
# MDN SERVER # OK    # connected
```

> ***Note:*** *"myDatalogAIR232 " is the title of the firmware that is installed on the rapidM2M module M11 . This type of site must then be created when generating the site on the myDataweb server (see " myDataweb-Server user manual " 206.886).*

The current firmware of the M11 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 48.

5.1.10 Button to reset the wireless GSM controller on the rapidM2M module M11 (S100)

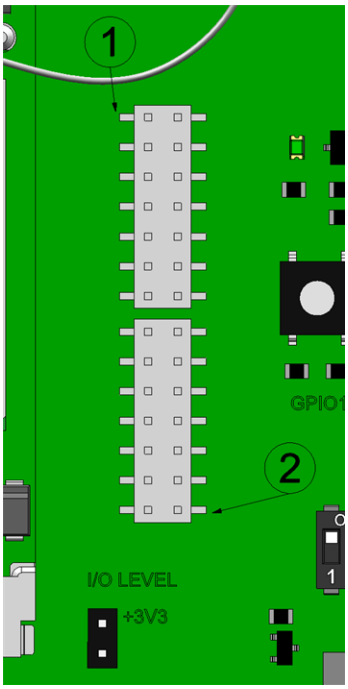
This button should only be used as an emergency reset, as the wireless GSM controller independently completes a reset process during the boot process. However, if a reset is still required, button S100 must be pressed briefly (200µs). The wireless GSM controller of the M11 module must then be switched back on via button S101 (see "Button to switch on the wireless GSM controller on the rapidM2M module M11 (S101)" on page 28).

A detailed description of the timings is provided in "Reset signal" on page 49.

5.1.11 Holding fixture for the M11 module

The holding fixture for the rapidM2M module M11 used on the rapidM2M Developer Board is manufactured by JAE Connectors. The manufacturer's designation for this connector is "MM60-52B1-G1-R850".

5.1.12 Plug connector to which the even-numbered pins of the M11 module are connected (J107 and J110)



Plug connector to which the even-numbered pins of the M11 module are connected

1	Pin 1	**2**	Pin 28

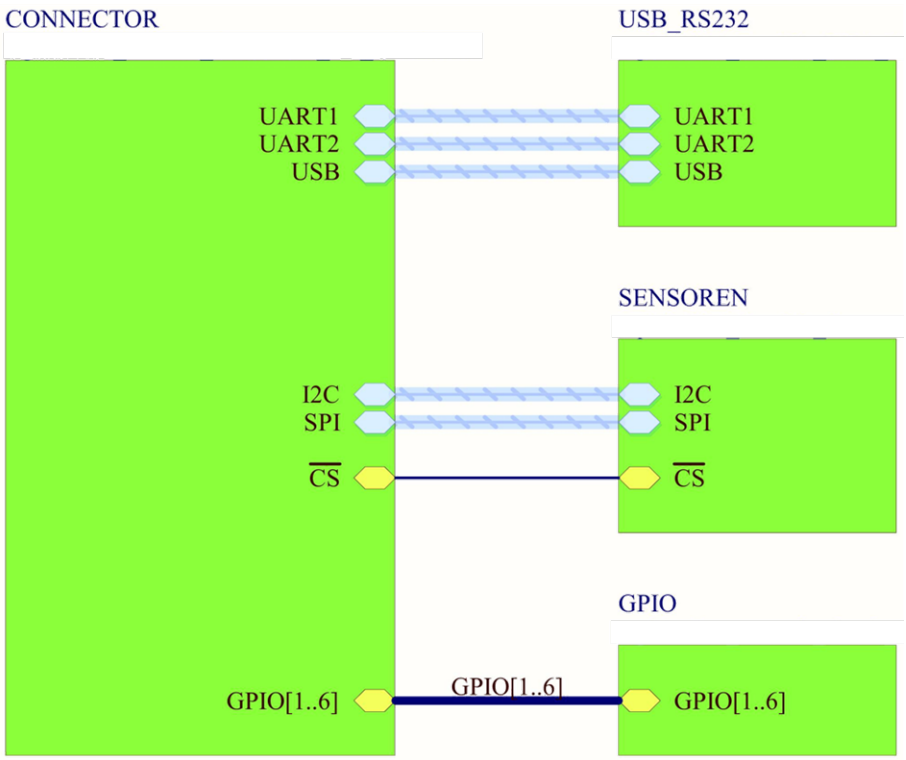
Pin	Type	Title	Description
1.27-28	I	V_{IN}	Supply voltage (4,5...32VDC) for the rapidM2M module M11
2.18	I/O	V_{AUX}	3,3V voltage output if the rapidM2M module M11 is supplied via V_{IN} Supply voltage input (3,2...4,5V) if no voltage is applied to V_{IN}
3-8			Reserved for extensions[1)]
9-10		GND	Mass
11-12			Reserved for extensions[1)]
13		GND	Mass
14		NC	
15-17			Reserved for extensions[1)]
19-20			Reserved for extensions[1)]
21-22		GND	Mass
23-26			Reserved for extensions[1)]

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

5.2 Circuit diagram

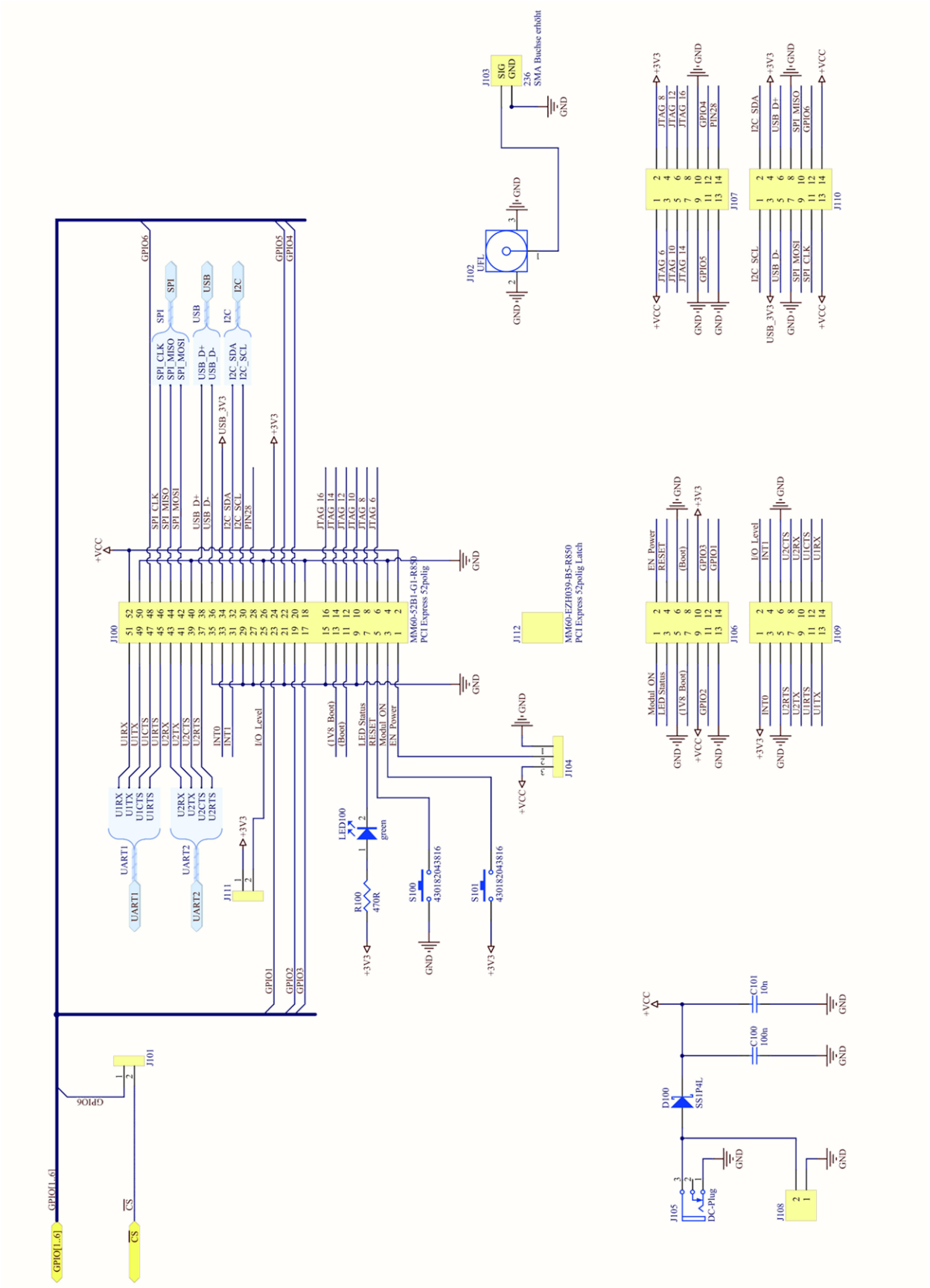
Note: *The following chapter includes the complete circuit diagram of the developer board. Some of the assemblies included are not supported by the rapidM2M module M11 and are provided for extensions.*

5.2.1 Overview

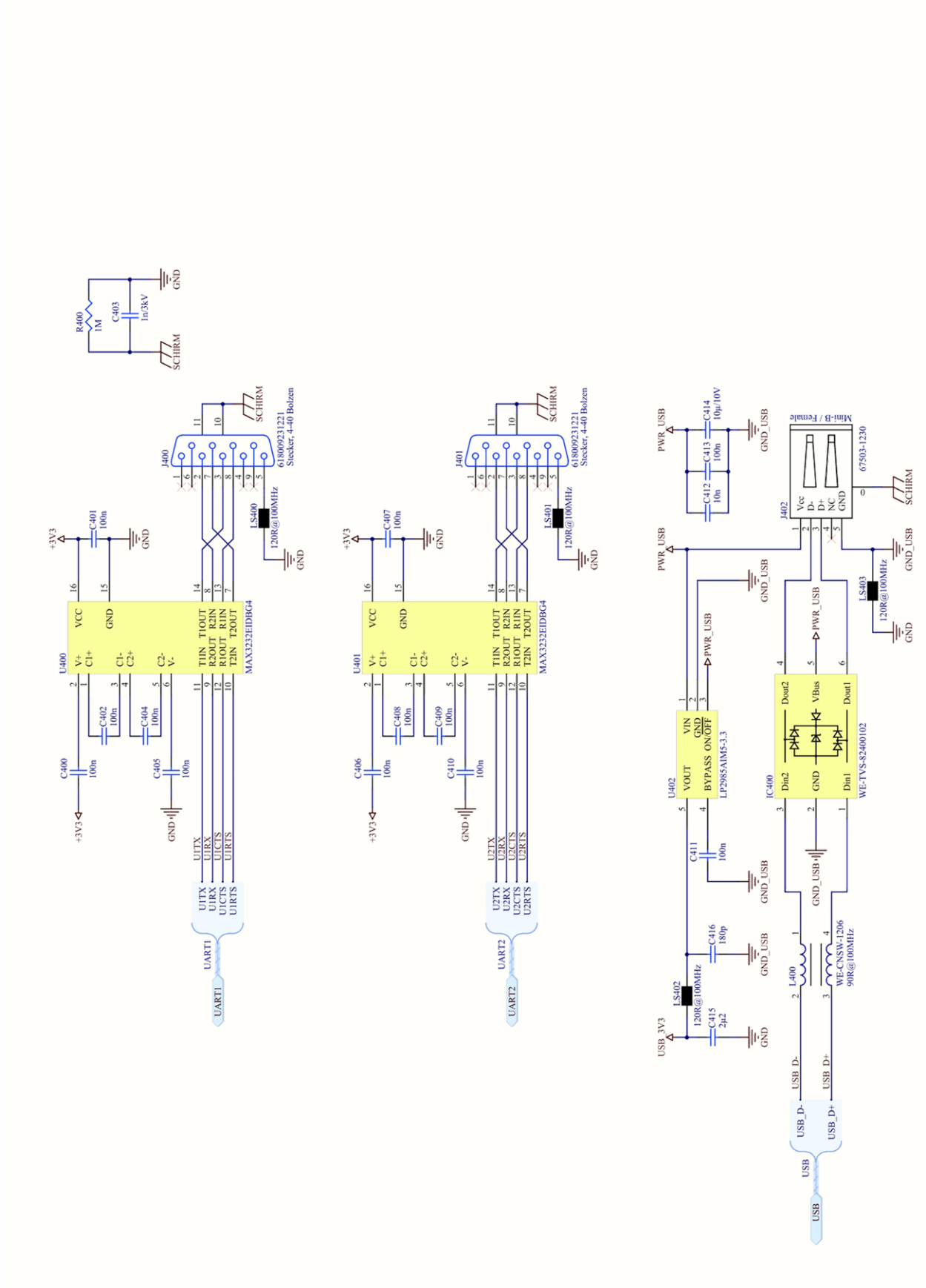


Overview of the circuit diagram of the developer board

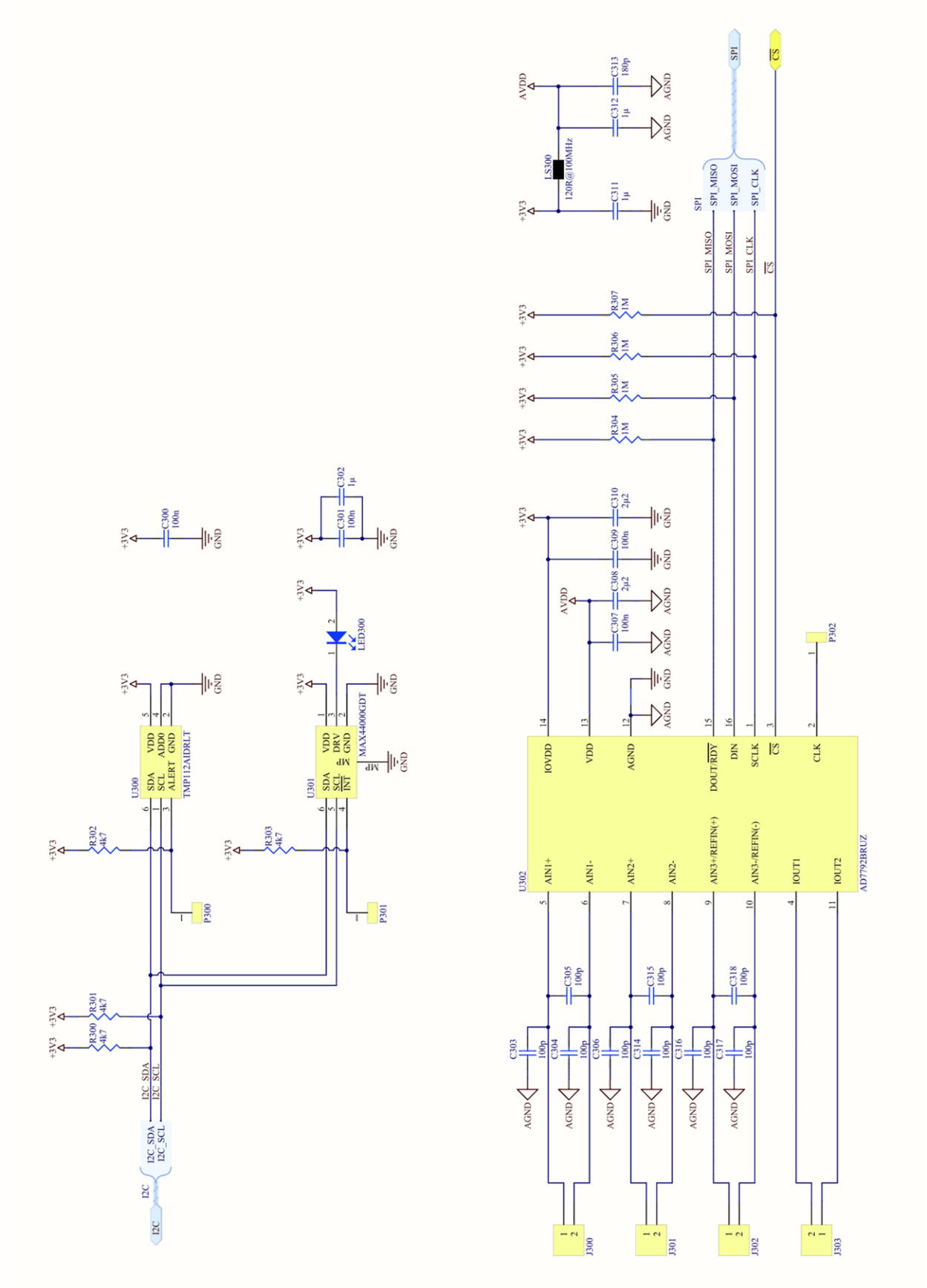
5.2.2 Connector



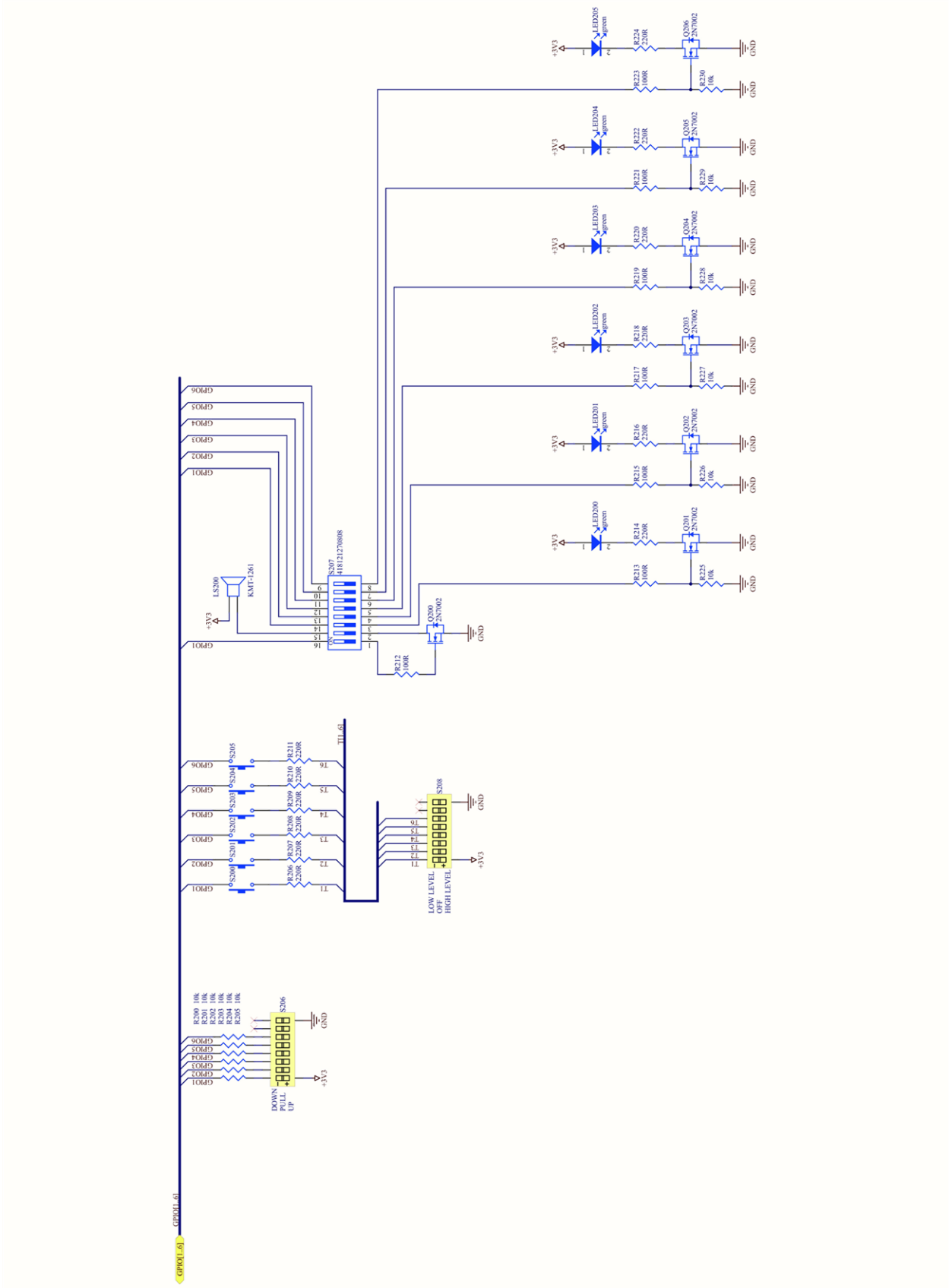
5.2.3 USB RS232



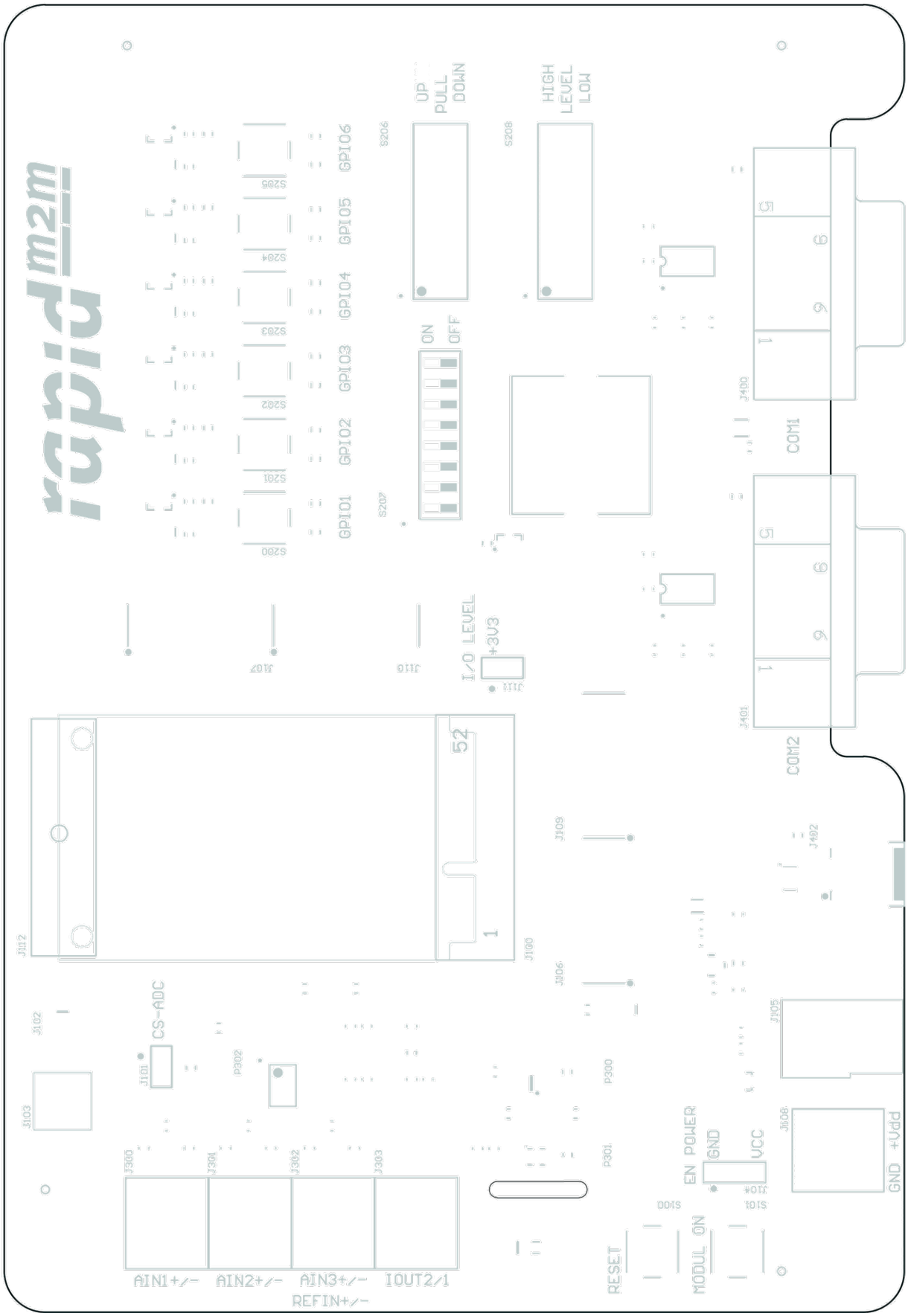
5.2.4 Sensors


U300
TMP112AIDRLT
U301
MAX44006GDT
LED300
P300
P301
U302
AD7792BRUZ
P302
J300
J301
J302
J303
I2C
SPI
CS
LS300
120R@100MHz

5.2.5 GPIO


LED205
LED204
LED203
LED202
LED201
LED200
Q206
Q205
Q204
Q203
Q202
Q201
Q200
2N7002
R224
R223
R222
R221
R220
R219
R218
R217
R216
R215
R214
R213
R212
R210
R209
R208
R207
R206
R211
R230
R229
R228
R227
R226
R225
220R
100R
10k
+3V3
GND
LS200
KMT-1261
S207
41812127808
S206
S208
S205
S204
S203
S202
S201
S200
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO1 61
TH 61
R200 10k
R201 10k
R202 10k
R203 10k
R204 10k
R205 10k
DOWN
PULL
UP
LOW LEVEL
OFF
HIGH LEVEL

5.3 Layout


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

5.4 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.

5.4.1 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.

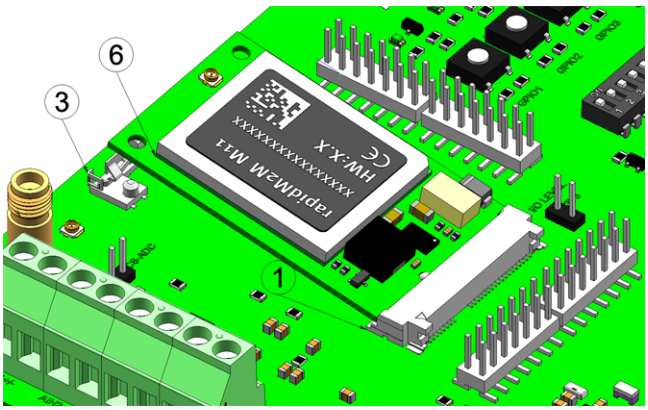
The sensitive electronic components inside the device can be damaged by static electricity, which can impair the device performance or even cause the device to fail. The manufacturer recommends the following steps to prevent any damage to the device caused by electrostatic discharges:

- 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.

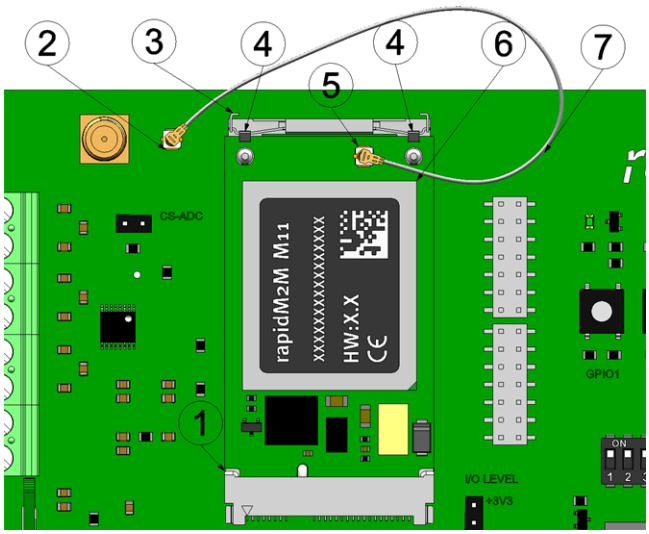
5.4.2 Inserting the M11 module

Important note:

- *The M11 module must be inserted in the de-energised state.*
- *Improper handling can lead to injuries and/or damage to the instruments!*



Inserting the M11 module - step 1



Inserting the M11 module - step 2

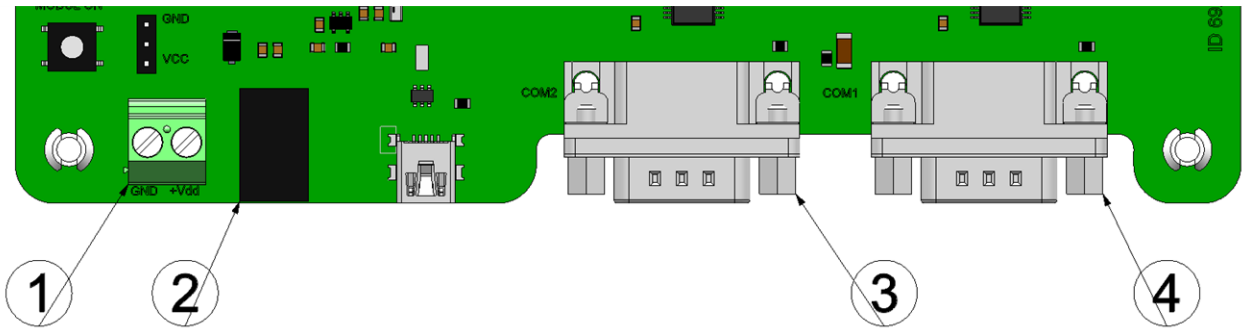
1	52pin plug connector (mechanically compatible with "PCI express mini card")	5	U.FL antenna connector of the rapidM2M module M11
2	U.FL antenna connector of the developer board (J102)	6	rapidM2M module M11
3	Locking device	7	U.FL antenna cable
4	Locking device hooks		

1. Insert the rapidM2M module M11 (6) in the 52pin plug connector (1) of the developer board in accordance with the Figure "Inserting the M11 module - step 1 " on page 38. 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 M11 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 M11 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.

5.4.3 Connection of the sensor and power supply

Important note:

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



Connection of the sensor and power supply

1	Power supply: Terminal strip (J108)	3	Sensor connector COM2: 9-pin D-SUB male (J400)
2	Power supply: DC coaxial connector (J105) 5,5/2,1mm (external/internal diameter)	4	Debug interface COM1: 9-pin D-SUB male (J401)

Assignment of the terminal strip (J108)

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

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

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

1. Firstly, connect the read-out digital sensor with the sensor connection (4) of the developer board. If possible, switch off the power supply for the digital sensor before establishing the RS232 connection between the rapidM2M Developer Board and the digital sensor.

   Note: *An adapter may be required for this purpose. A selection of available adapters is specified in the chapter "Spare parts and accessories" (see "Spare parts and accessories" on page 137).*

   Important note: *The rapidM2M module M11 included in the scope of supply of the developer package does not support a hardware handshake.*

2. Connect the antenna (see "Connecting the GSM antenna" on page 40).

3. The rapidM2M Developer Board can be powered via the DC coaxial connector (2) or the terminal strip (1). Either use the plug-in power supply included in the scope of delivery or any power supply unit with a voltage of 5...32 VDC . If the developer board is to be powered via the terminal strip (1), the cables must be connected to the terminal strip (1) in a de-energised state.
4. 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 (see "Overview" on page 21).
5. Now switch on the power supply of the digital sensor and developer board.
6. Then press the button to switch on the wireless GSM controller S101 (see "Button to switch on the wireless GSM controller on the rapidM2M module M11 (S101)" on page 28). The status LED should then light up (see "Status LED (LED100)" on page 27) to indicate that the rapidM2M developer package M11 is logging into the network and is establishing a connection with myDataweb.

   Note: *Additional status information is available via the debug interface COM1 (J401).*

The following step is not mandatory.

7. Check whether the connection to the myDataweb has worked correctly (see "Testing communication with the device" on page 53).

5.4.4 Connecting the GSM antenna

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

The Magnetic roof antenna SMA (300085) included in the scope of supply is connected directly to the antenna connector (see "Overview" on page 21) of the developer board.

If the 2.5 m connection cable of the Magnetic roof antenna SMA (300085) is not long enough, you can use a 2.5 m Extension for ANT900 SMA (206.807).

1. Ensure that the rapidM2M Developer Board is de-energised.
2. If you need an antenna extension, connect this to the antenna first.
3. Connect the antenna extension or antenna directly to the antenna connector of the developer board (see "Overview" on page 21).

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

4. Switch the power supply of the developer board back on.
5. Then press the button to switch on the wireless GSM controller S101 (see "Button to switch on the wireless GSM controller on the rapidM2M module M11 (S101)" on page 28).

The following step is not mandatory.

6. Check whether the connection to the myDataweb has worked correctly (see "Testing communication with the device" on page 53).

6872
6872
6872

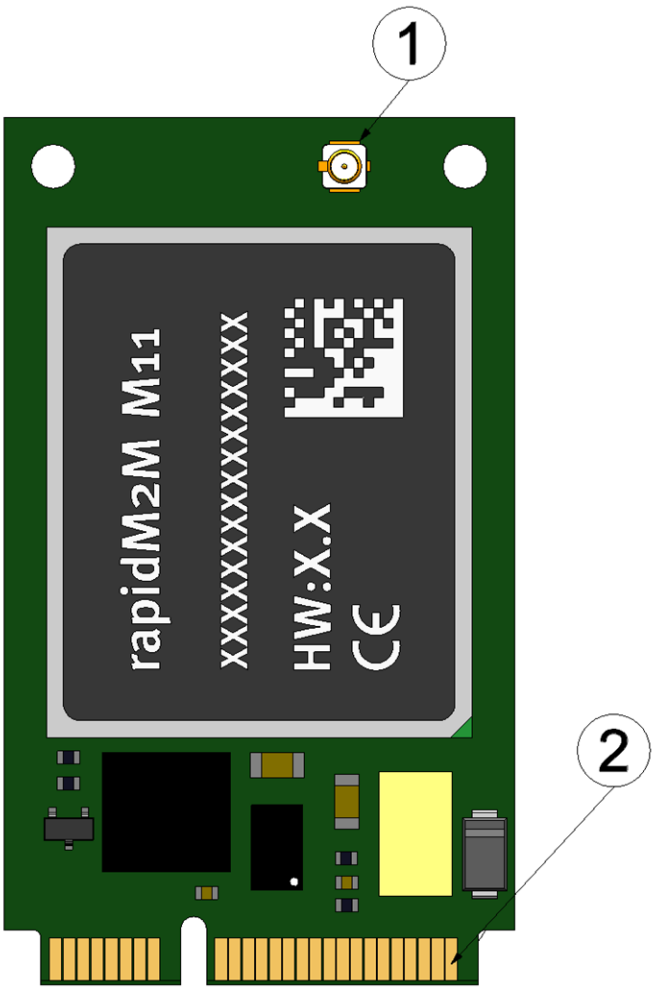
Chapter 6 rapidM2M module M11

6.1 Technical data

Voltage supply	Supply via V_{IN}: 4,5...32VDC , max. t.b.d. mA Supply via V_{AUX}: 3,2...4,5V , max. t.b.d. mA Additional information is provided in "Technical details regarding the energy supply" on page 48.
Output voltage (only when supplied via V_{IN})	3,3V , 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 50.
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 27.
Antenna connector	U.FL
Interfaces	1x UART for the connection of a digital sensor • Baud rate: 300-115200 • Stop bits: 1,2 • Parity:N, E, O • Data bits: 7, 8 Modes: • ASCII 1x UART for the output of debug information Additional information is provided in "Technical details regarding UART1" on page 50 and "Technical details regarding UART2" on page 51.

Data memory	Internal flash memory for up to 2620 measurement cycles
Data type	f32 (32 Bit floating point)
Data transmission	By means of GSM/GPRS quad-band modem to the respective myDataweb server
SIM	The rapidM2M module M11 is equipped with an integrated SIM chip.
Monthly data volume	t.b.d. at two-minute measurement cycle and 120-minute transmission cycle

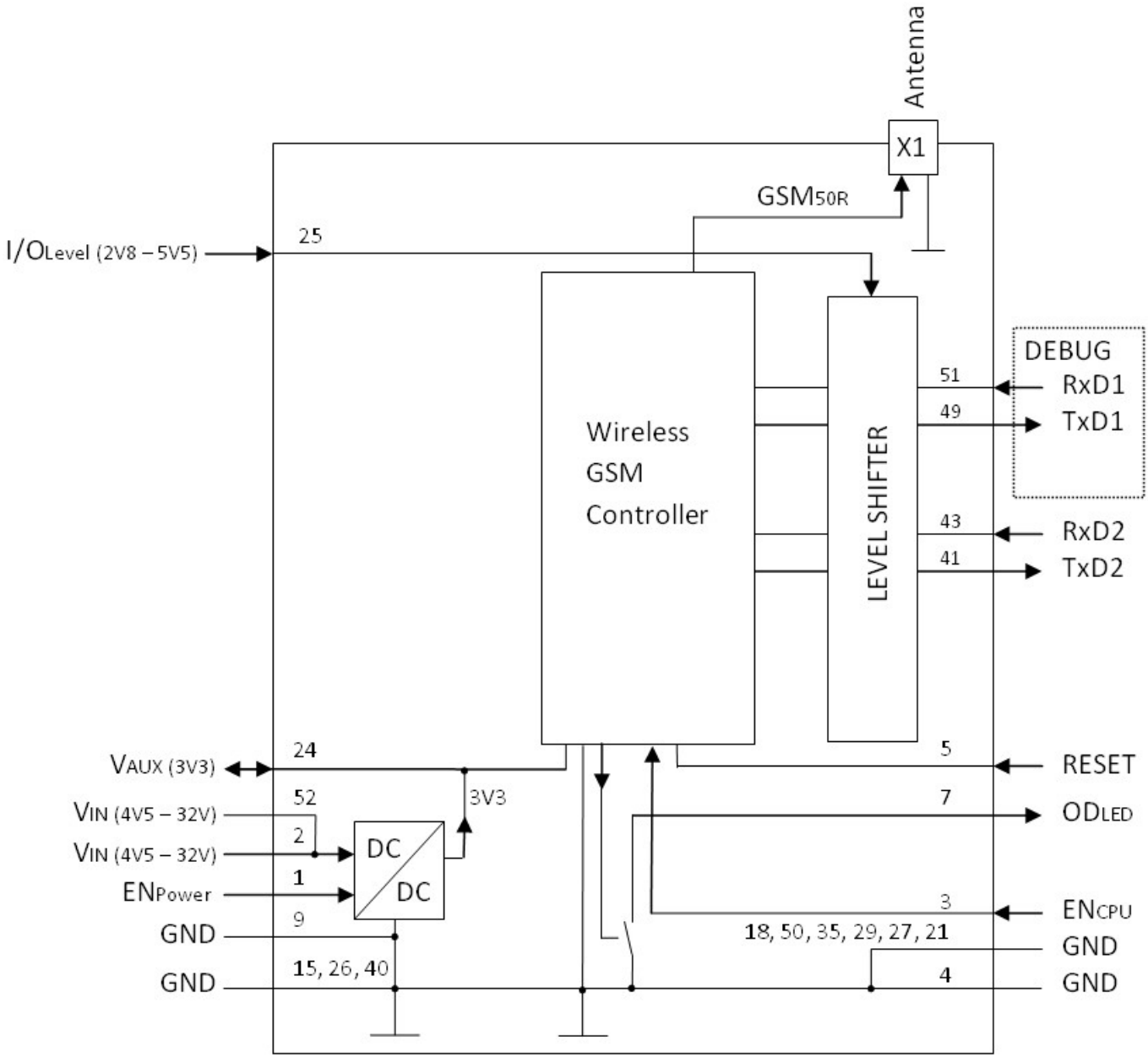
6.2 Overview



Overview rapidM2M module M11

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

6.2.1 Block diagram



Block diagram of the M11 module

6.2.2 52pin connector strip (X2)

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 rapidM2M module M11
3	I	EN_{CPU}	Pin to switch on the wireless GSM controller on the rapidM2M module M11
4, 9, 15, 18, 21, 26, 27, 29, 35, 40, 50		GND	Mass
5	I	Reset	Pin to reset the wireless GSM controller on the rapidM2M module M11
6, 8, 10, 12, 14, 16, 28		NC	
7	O	OD_{LED}	Open drain output to connect a status LED
11, 13, 17, 19, 20, 22, 23, 30-34, 36-39, 42, 44-48			Reserved for extensions No signals must be connected to these pins.
24	I/O	V_{AUX}	3,3V voltage output if the rapidM2M module M11 is supplied via V_{IN}. Supply voltage input (3,2...4,5V) if no voltage is applied to V_{IN}.
25	I/O	I/O level	2,8...5,5V supply of the level shifters to adjust the signal level of the M11 module to the surrounding hardware
41	O	TxD2	TxD line of COM2
43	I	RxD2	RxD line of COM2
49	O	TxD1	TxD line of COM1
51	I	RxD1	RxD line of COM1

6.2.3 Device labelling

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

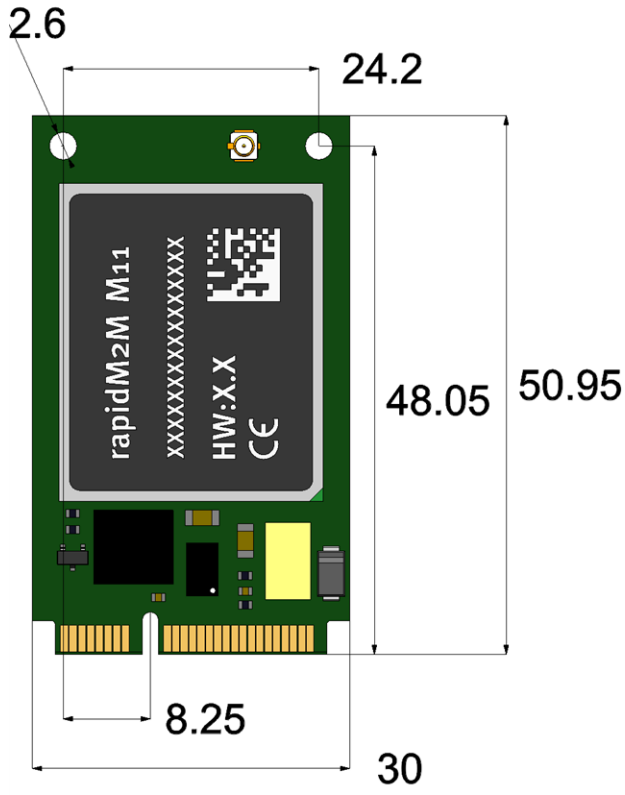
- Type designation
- Serial number
- CE marking
- Hardware revision

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.

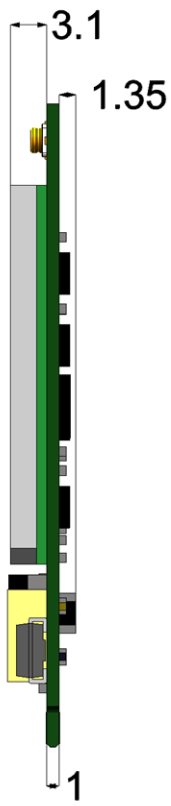


Type plate rapidM2M module M11

6.3 Dimensions

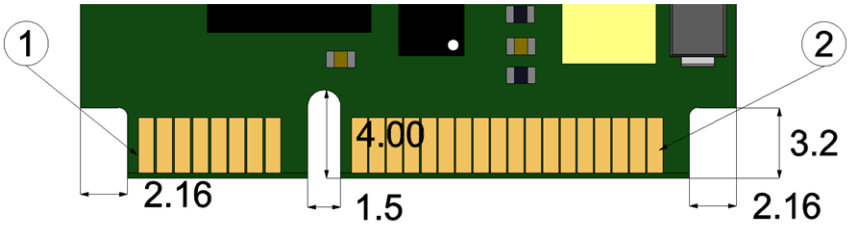


Dimensions: Width and depth (view from above)



Dimensions: Height

6.3.1 Dimensions of the 52pin connector strip (X2)



Detailed overview of the connector strip (from above)

1 Pin 1	2 Pin 51

6.4 Technical details

6.4.1 Technical details regarding the energy supply

Depending on the voltage available, there are two options to supply the rapidM2M module M11 :

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

6.4.1.1 Supply via V_{IN}

The internal supply voltage (3,3V) is generated by the DC/DC converter of the M11 module (see "Block diagram" on page 45) . 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. It therefore takes 50ms until the internal supply voltage has reached its maximum value. The "V_{AUX}" pin functions as a voltage output for the internal 3,3V supply voltage if the rapidM2M module M11 is supplied via V_{IN}.

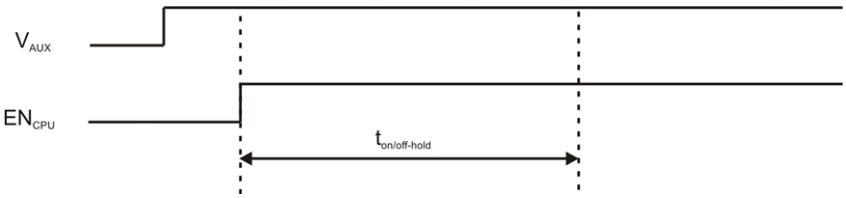
6.4.1.2 Supply via V_{AUX}

The DC/DC converter of the M11 module (see "Block diagram" on page 45) is omitted and the supply voltage is fed directly in to the module via the "V_{AUX}" pin.

6.4.2 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. ($t_{on/off-hold}$) 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 rapidM2M module M11 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 M11 module does not provide the means to switch off the wireless GSM controller.

6.4.3 Reset signal

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

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 48). 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.).

6.4.4 Status LED

The rapidM2M module M11 is equipped with an open drain output for connecting a status LED ("OD_{LED}" pin) to signal the operating state.

Operating states

Status LED	Description
Off	The rapidM2M module M11 is not connected to the power supply.
Lights up	The rapidM2M module M11 is connected to the power supply but has not yet been logged into the network.
Flashes t_{on}/t_{off}: 0,2/2sec.	The rapidM2M module M11 is logged into the network.
Fast flashing t_{on}/t_{off}: 0,2/0,6sec.	The rapidM2M module M11 is connected to the myDataweb server. (e.g. online mode or ALOHA transmission mode)

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

6.4.5 Technical details about the level shifters

Parameter	min.	typ.	max.
$V_{I/O\ Level}$	2,8V		5,5V
V_{IL}			0,15V
V_{IH}	$V_{I/O\ Level}$ - 0,4V		

The module is equipped with level shifters for all of the I/O signals so that the signal level of the M11 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 rapidM2M Module M11 rapidM2M module M11 on the "V_{AUX}" pin can be fed in via the "I/O level" pin (see "Block diagram" on page 45).

6.4.6 Technical details regarding UART1

> ***Important note:*** *The UART1 of the M11 module is designed for the output of debug information, whereby the hardware handshake is not active. The UART1 cannot be used to record data.*

The UART1 level of the M11 module is dependent on the supply voltage supplied at the "I/O level" pin for the level shifters (see "Block diagram" on page 45).

Baud rate	115200
Stop bits	2
Parity	N
Data bits	8
Modes	ASCII

6.4.7 Technical details regarding UART2

> ***Important note:*** *The rapidM2M module M11 included in the scope of supply of the developer package does not support a hardware handshake.*

The UART2 level of the M11 module is dependent on the supply voltage supplied at the "I/O level" pin for the level shifters (see "Block diagram" on page 45).

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

6.4.7.1 Error codes of the UART2 interface

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
2000	MODULE ERR	0	CHAR TIMEOUT	The time lag between two received characters is larger than the character timeout entered via the input screen for configuring the interface (see "Serial interface" on page 60).
		1	FRAME TIMEOUT	The complete answer was not received within the frame timeout. This period of time is entered via the input screen for configuring the interface (see "Serial interface" on page 60).
		2	INVALID FRAME	The answer frame of the sensor is invalid if, for example, the number of columns of the sensor does not correspond to the number entered via the configuration interface (see "Column" in the chapter "Interface channels 1-32" on page 56).

6872
6872
6872

Chapter 7 Placing into operation

7.1 Placing the system into operation

1. Once you have completed all of the steps described in chapter "Connection of the sensor and power supply" on page 39, the rapidM2M module M11 is ready for operation and should have already made its first connection to the myDataweb server.
2. Create a site for the operation on the myDataweb server (see "myDataweb-Server user manual " 206.886).

   Note: *A "myDatalogAIR232 " type site must be created to operate the M11 module.*
3. Configure the created site according to your requirements (see "Site configuration" on page 55).
4. Link the rapidM2M module M11 with the created site (see "Site" on page 55).
5. Initiate the ALOHA transmission mode via the myDataweb server (see "Basic settings" on page 66) so that the configuration of the site is transferred to rapidM2M module M11 . This is only possible if the "interval and wakeup" connection type has been selected. If "interval" is selected as the connection type, either wait until the next scheduled connection or disconnect the rapidM2M module M11 from the supply voltage and then press the button to switch on the wireless GSM controller once the supply voltage is restored (see "Button to switch on the wireless GSM controller on the rapidM2M module M11 (S101)" on page 28). With the "online" connection type, the configurations are immediately transmitted to rapidM2M module M11 following input on the myDataweb server.

7.2 Testing communication with the device

1. Create a site for the operation on the myDataweb server (see "myDataweb-Server user manual " 206.886).
2. Configure the created site according to your requirements (see "Site configuration" on page 55).
3. Link the rapidM2M module M11 with the created site (see "Site" on page 55).
4. Initiate the ALOHA transmission mode via the myDataweb server (see "Basic settings" on page 66) so that the configuration of the site is transferred to rapidM2M module M11 . This is only possible if the "interval and wakeup" connection type has been selected.
5. Wait until it is indicated in the list of measurement instruments that the device is in ALOHA transmission mode. This is indicated by a speech bubble with the "Aloha" inscription.



6. Check the "GSM level" in the ALOHA data window of the myDataweb server, which can be accessed by clicking on the speech bubble with the "Aloha" inscription (see "myDataweb-Server user manual " 206.886).

   Note: *Supplementary explanation on evaluating the "GSM level":*

"GSM level"	
›-64dBm	
-64 to -73dBm	
-74 to -83dBm	
-84 to -93dBm	
-94 to -107dBm	
‹= -108dBm	

Chapter 8 User interface on the myDataweb-Server

8.1 Site configuration

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

Click on the name of the site in the list of sites to get to the input screen for configuring the site (see "myDataweb-Server user manual " 206.886).

8.1.1 Site

Customer
Specifies to which customer the site is assigned.

 **icon**
Assign site to another customer

Name
Site designation (not relevant for the device or data assignment)

Instrument S/N
Serial number of the device that is linked to the site (device assignment!)

8.1.2 Comments

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

8.1.3 Control

Script type	Off	Script deactivated
	IL	Activates script processing and informs the rapidM2M developer package M11 that the script is an "Instruction list".
	Pawn	Activates script processing and informs the rapidM2M developer package M11 that the script is a "Pawn script".
Script template	"Name of a template"	List of the script templates compatible with the script type selection that have been created for the rapidM2M developer package M11 on the myDataweb server (to create a template, see "myDataweb-Server user manual " 206.886). The template is copied into the input window to edit the script and can also be adapted before it is loaded into the device.
Script	Input window for editing the script that is loaded in rapidM2M developer package M11 .	

8.1.4 Interface channels 1-32

8.1.4.1 Basis

Title 1-32

Freely selectable channel title for the interface channels

Interface

Basic setting for the interface channel

Off	---	*Interface channel deactivated*
RS232	*Column*	*The ASCII data message from the digital sensor is split into individual measurement values by the "number delimiter" (see "Serial interface" on page 60). The "column" configuration parameter specifies which of these measurement values is linked with/recorded on the interface channel.*
	Min	*Defines the lower scale end of the pointer instruments*
	Max	*Defines the upper scale end of the pointer instruments*
	Unit	*String, which is used as the measurement unit by all of the server display elements*
	Decimal places	*Number of decimal places that are used by all of the server display elements*

8.1.4.2 Alarms

Warning	Value low	A warning is initiated if the measurement value drops to or below this value.
	Value high	A warning is initiated if the measurement value meets or exceeds this value.
Alarm	Value low	An alarm is initiated if the measurement value drops to or below this value.
	Value high	An alarm is initiated if the measurement value meets or exceeds this value.
Hyst %	Hysteresis for all-clear (e.g. hyst = 5%, alarm or warning at 100 -› all-clear at 95)	

8.1.4.3 Trigger

QU	Fastrecording (recording cycle = recording cycle / factor)	
SL	Slow recording (recording cycle = recording cycle * factor)	
RO	Switch on recording	
RF	Switch off recording	
XM	Initiate transmission	
ON	Activate online mode	
Level	Levels for initiating the trigger	
	Greater or equal	The measurement value must be greater than or equal to the level to initiate the trigger.
	Less or equal	The measurement value must be less than or equal to the level to initiate the trigger.

8.1.5 Interface channels 33-64

8.1.5.1 Basis

Title 33-64

Freely selectable channel title for the interface channels

The significance of the remaining configuration parameters in this configuration section corresponds to the "Interface channels 1-32" configuration section (see "Interface channels 1-32" on page 56).

8.1.6 Interface output channels 1-32

Title 1-32

Freely selectable channel title for the output channel

Mode

Basic setting for the output channel

Off	*Output channel deactivated*	
On	*Output channel activated*	
	Setpoint	*Setpoint that should be issued*

Note: *If one or more output channels are active, the rapidM2M developer package M11 imports and processes the ASCII data message from the digital sensor and then issues an ASCII data message including the setpoints entered via the input screen on the myDataweb server or calculated by means of the instruction list (see "Instruction list" on page 71). During this process, the same frame formats are used that the rapidM2M developer package M11 would expect from the digital sensor. The first number delimiter specified in the "Serial interfaces" section is used as the number delimiter (see "Serial interface" on page 60).*

8.1.7 Interface output channels 33-64

Title 33-64

Freely selectable channel title for the output channel

The significance of the remaining configuration parameters in this configuration section corresponds to the "Interface output channels 1-32" configuration section (see "Interface output channels 1-32" on page 57).

8.1.8 Calculated channels

Note: *The values of the calculated channels are calculated directly for the data output (display on myDataweb server or download from myDataweb server). They are not saved in the server database.*

8.1.8.1 Basis

Title 1-5

Freely selectable channel title for the calculated channels

Mode

Possible calculation modes for the calculated channels

Off	*---*	*Calculated channel deactivated*
Table	*Min*	*Defines the lower scale end of the pointer instruments*
	Max	*Defines the upper scale end of the pointer instruments*
	Unit	*String, which is used as the measurement unit by all of the server display elements*
	Decimal places	*Number of decimal places which are used by all of the server display elements*
Digital	*Invert*	*Inverts the input signal*
+, -, x, /	*Min*	*Defines the lower scale end of the pointer instruments*
	Max	*Defines the upper scale end of the pointer instruments*
	Unit	*String, which is used as the measurement unit by all of the server display elements*
	Decimal places	*Number of decimal places which are used by all of the server display elements*



Shifts element down



Shifts element up

8.1.8.2 Calculation

Off	---	Calculated channel deactivated
Table	Source	Selection of the channel from which the input data is used
		Opens the screen for entering the values table (between the table rows is interpolated linearly, values outside of the defined table are extrapolated linearly.)
Digital	Source	Selection of the channel from which the input data is used
	High level	Signal recognition level
+, -, x, /	Source	Selection of the channel from which the input data is used
	+, -, x, /	
	Source	Selection of the channel from which the input data is used

8.1.8.3 Alarm

Note: *The alarm levels for calculated channels can only be evaluated once the device has transmitted the measurement data to the myDataweb server.*

Alarm low	An alarm is initiated if the measurement value drops to or below this value.
Alarm high	An alarm is initiated if the measurement value meets or exceeds this value.
Hyst %	Hysteresis for all-clear (e.g. hyst = 5%, alarm or warning at 100 -› all-clear at 95)

8.1.9 Internal channels

8.1.9.1 Basis

GSM level title	Freely selectable channel title for the GSM field level	
	Unit	String, which is used as the measurement unit by all of the server display elements

8.1.9.2 Alarms

Warning	Value low	A warning is initiated if the measurement value drops to or below this value.
	Value high	A warning is initiated if the measurement value meets or exceeds this value.
Alarm	Value low	An alarm is initiated if the measurement value drops to or below this value.
	Value high	An alarm is initiated if the measurement value meets or exceeds this value.
Hyst %	Hysteresis for all-clear (e.g. hyst = 5%, alarm or warning at 100 -› all-clear at 95)	

8.1.9.3 Trigger

QU	Fastrecording (recording cycle = recording cycle / factor)	
SL	Slow recording (recording cycle = recording cycle * factor)	
RO	Switch on recording	
RF	Switch off recording	
XM	Initiate transmission	
ON	Activate online mode	
Level	Levels for initiating the trigger	
	Greater or equal	The measurement value must be greater than or equal to the level to initiate the trigger.
	Less or equal	The measurement value must be less than or equal to the level to initiate the trigger.

8.1.10 Serial interface

Profile

Basic settings for the serial interface

Standard (1/3)	*Baud rate*	*300*	*Selection of the required baud rate*
		600	
		1200	
		2400	
		4800	
		9600	
		19200	
		38400	
		57600	
		115200	
	Stop bits	*1*	*Selection of the required stop bits*
		2	
	Parity	*None*	*Selection of the required parity*
		Odd	
		Even	
	Data bits	*7*	*Number of data bits to be used*
		8	

Standard (2/3)	*Retry*	*In the event of a communication error, the corresponding command is repeated once. The error is only indicated if this second attempt also fails.*	
	Frame structure	*...[CR][LF]*	*Each frame must be terminated with both of the "[CR][LF]" characters.*
		...[ETX]	*Each frame must be terminated with the "[ETX]" character.*
		[STX]...[CR][LF]	*Each frame must start with the "[STX]" character and terminate with both of the "[CR][LF]" characters.*
		[STX]...[ETX]	*Each frame must start with the "[STX]" character and terminate with the "[ETX]" character.*
	Frame timeout	*0 to no timeout* *Time during which the connected digital sensor must send the full answer to a request. A communication error is detected if the time is exceeded.*	
	character timeout	*0 to no timeout* *Maximum permissible time interval between the full receipt of two characters. A communication error is detected if the time is exceeded. The monitoring process starts following receipt of the first character of an answer from the digital sensor.*	
	Number delimiter	*Divides the ASCII data message of the digital sensor into individual measurement values. Up to ten different delimiters can be entered via the input field. The following escape codes are also supported as delimiters:* *\a \b \f \n \r \t \v \\ \? \' \" \xhh* *The maximum number of delimiters is reduced when escape codes are used.*	
	Number format	*1,000.00*	*"," is used as the thousand separator* *"." is used as the decimal point*
		1,000.00	*"." is used as the thousand separator* *"," is used as the decimal point*
	Warm-up time	*Time from switching on the rapidM2M developer package M11 until the init sequence is sent.*	

Standard (3/3)	*Hold*	*Maintain the last valid measurement value for x measurement cycles*	
		off	*Function deactivated*
		1-5	*Number of measurement cycles for which the measured value is maintained until the error value is issued*
		on	*In the event of an error, the last valid measurement value is maintained until a new valid measurement value is present.*
	Init sequence	*Initialisation command that is sent to the digital sensor when the rapidM2M developer package M11 is switched on following expiry of the warm-up time.* *The following escape codes are supported:* *\a \b \f \n \r \t \v \\ \? \' \" \xhh*	
	Measurement data sequence	*Command that is sent to the digital sensor at every measuring point.* *The following escape codes are supported:* *\a \b \f \n \r \t \v \\ \? \' \" \xhh*	

YSI 6 series	*Presetting for the YSI 6 series multi-parameter water quality measurement sensors*		
	Baud rate	*300*	*Selection of the required baud rate*
		600	
		1200	
		2400	
		4800	
		9600	
		19200	
		38400	
		57600	
		115200	
	Retry	*In the event of a communication error, the corresponding command is repeated once. The error is only indicated if this second attempt also fails.*	
	Hold	*Maintain the last valid measurement value for x measurement cycles*	
		off	*Function deactivated*
		1-5	*Number of measurement cycles for which the measured value is maintained until the error value is issued*
		on	*In the event of an error, the last valid measurement value is maintained until a new valid measurement value is present.*

ISCO periodic update	*Presetting for an ISCO sampling device*		
	Baud rate	*300*	*Selection of the required baud rate*
		600	
		1200	
		2400	
		4800	
		9600	
		19200	
		38400	
		57600	
		115200	
	Retry	*In the event of a communication error, the corresponding command is repeated once. The error is only indicated if this second attempt also fails.*	
	Timeout data set	*0 to no timeout* *Time during which the connected digital sensor must send the full answer to a request. A communication error is detected if the time is exceeded.*	
	Hold	*Maintain the last valid measurement value for x measurement cycles*	
		off	*Function deactivated*
		1-5	*Number of measurement cycles for which the measured value is maintained until the error value is issued*
		on	*In the event of an error, the last valid measurement value is maintained until a new valid measurement value is present.*

8.1.11 Alarm settings

Acknowledgement	Standard	The global server setting is used (see "myDataweb-Server user manual " 206.886) to determine whether to acknowledge alarms 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.
Offline alarm	Alarm in the event that the device does not respond for longer than the set transmission cycles	
Transfer volumes	Standard	The configuration for the transfer volume alarm is adopted from the global server setting (see "myDataweb-Server user manual " 206.886).
	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.
On alarm	A	The alarm is recorded in the alarm list.
	Ü	An immediate transmission is initiated.
On warning	A	The alarm is recorded in the alarm list.
	Ü	An immediate transmission is initiated.
On fault alarm	A	The alarm is recorded in the alarm list.
	Ü	An immediate transmission is initiated.
On fault warning	A	The alarm is recorded in the alarm list.
	Ü	An immediate transmission is initiated.
Test alarm	A test message is sent to all recipients ("parallel" alarm call plan type) and/or the first recipient ("serial" alarm call plan type) as soon as this page is saved. The check box is subsequently reset automatically.	

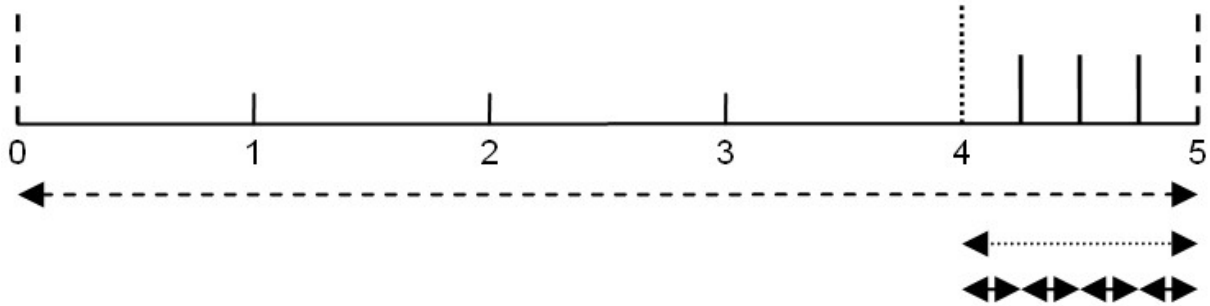
8.1.12 Basic settings

Connection type	Cycle	The device registers in the transmission cycle
	Cycle & wakeup	The device registers in the transmission cycle and can be placed into the ALOHA transmission mode via the server. 
	Online	The device does not disconnect the connection and continuously transmits the measurement data.
Aloha/wakeup duration	Aloha/wakeup duration connection	
Transmission cycle	Time between transmissions	
Recoding cycle	Time between measurement data recordings	
Measure quick divisor	Recoding cycle = recording cycle / factor (from triggering)	
Measure slow factor	Recoding cycle = recording cycle * factor (from triggering)	
Measurement cycle	Time between measurements (00:00 same as recoding cycle)	
Burst cycle	Period of time, during which measurement intervals are used for measuring, before recording begins. (00:00 measurement cycle is permanently activated)	
Time zone	Regional setting (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 "myDataweb-Server user manual " 206.886).
	Off	Automatic time adjustment deactivated
	USA	Predefined setting for the American area
	EU	Predefined setting for the European area
Position cycle	Time between position update	
Default view	Report, which is loaded into the map by clicking on the device link	

Note:

Example for clarification of the recoding, measurement and burst interval

Basic setting		*Recoding interval*	*5 min.*
		Burst interval	*1 min.*
		Measurement cycle	*15 sec.*



Note: *Additional explanation of the connection types*

Connection type	*Energy consumption*	*Data volumes*	*Response time*
Online			
Cycle & wakeup			
Cycle			

8.1.13 FTP export settings

Note: *This configuration section is only visible if the "FTP Agent Extended" licence for the myDataweb-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 myDataweb-Server (for creating an FTP export profile, see "myDataweb-Server user manual " 206.886).
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	

8.2 Device configuration

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

You can reach the input screen for configuring the device by clicking on the serial number in the site list (see "myDataweb-Server user manual " 206.886) or by clicking on the device name in the device name list (see "myDataweb-Server user manual " 206.886).

8.2.1 Comments

Comments

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

8.2.2 Measurement instrument

Customer	Name of the customer to whom the measurement instrument is assigned
IMSI	IMSI of the used SIM chip or SIM card
Serial number	Serial number of the instrument
Instrument class	The instrument class of the site and instrument must match for an instrument to be able to be connected to a site.
Telephone number	Telephone number of the SIM card. The control SMS messages (e.g. wakeup) are sent to this number. Format: +43555837465
Instrument flags	Additional information on the instrument class (for internal use)
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 transmitted 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".
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).

8.2.3 Device-specific settings

Operating type	HOLD	Measurement: off, transfer: on
	RUN	Measurement: on, transfer: on
	Aloha	Reserved for internal use

8.2.4 GPRS

SIM tariff

Selected SIM tariff

6872
6872
6872

Chapter 9 Instruction list

9.1 General

The following chapter describes the functionality of the instruction list parser in accordance with IEC 61131-3:2003.

There are two ways to create an instruction list for the rapidM2M developer package M11 :

- Direct entry in the "Script" input field in the "Control" configuration section
- Use of a script template, that was created on the myDataweb server

9.1.1 Direct entry of an instruction list

The instruction list is entered via the "Control" configuration section (see "Control" on page 55) of the input screen for configuring the site. "IL" must be selected as the "script type" so that rapidM2M developer package M11 interprets the commands entered in the "script" as an instruction list.

9.1.2 Using a script template

Predefined script templates can be selected via the "Control program" list selection in the "Control" configuration section (see "Control" on page 55) of the input screen for configuring the site (to create a template, see "myDataweb-Server user manual " 206.886). After selecting the control program, the template is copied to the "Script" input window where is can be adapted. When using this method, "IL" must also be selected as the "script type" so that rapidM2M developer package M11 interprets the commands as an instruction list.

> ***Note:*** *If an instruction list is used to write directly to an output, the setpoint entered via the input screen of the myDataweb is overwritten with the calculated value.*

9.2 Implementation details

9.2.1 Common elements

No. of the table	Attribute	Description	Example/explanation
1	2	Lower case letters	
	3a	Hash (#)	
2	1	Capital letters and numbers	IW215 IW215Z QX75 IDENT
3	1	Comments	(**********) (* A framed comment *) (**********)
4	1	Integer literal	-12 0 123_456 +986
	2	Real literal	-12.0 0.0 0.456 3.14159_26
	7	Boolean zero and one	0 1
7	1a	Duration without underscore: short prefix	T#14ms T#-14ms T#14.7s T#14.7m T#14.7h t#14.7d t#25h15m t#5d14h12m18s3.5ms
8	6	Literal for date and time (short prefix)	DT#1984-06-25-15:36:55 dt#1984-06-25-15:36:55
10	1	BOOL	Data type: Boolean, 1 bit
	4	DINT	Data type: double integer, 32 bit
	8	UDINT	Data type: unsigned double integer, 32 bit
	10	REAL	Data type: real number, 32 bit
	12	TIME	Data type: Duration
	15	DATE_AND_TIME or DT	Data type: Date and time
	19	DWORD	Data type: Bit sequence 32, 32 bit
15	1	I (storage location input)	%IW215 %IX2.5
	2	Q (storage location output)	%QX75 %QR1
	3	M (storage location marker)	%MD48 %MR1000
	4	X ((individual) bit size)	
	5	No ((individual) bit size)	
	6	B (byte (8 bits) size)	
	7	W (word (16 bits) size)	
	8	D (double word (32 bits) size)	

No. of the table	Attribute	Description	Example/explanation
23	1	ABS	Absolute
	2	SQRT	Square root LD %IRo SQRT
24	12	ADD	OUT := IN1 + IN2
	13	MUL	OUT := IN1 * IN2
	14	SUB	OUT := IN1 - IN2
	15	DIV	OUT := IN1 / IN2
	16	MOD	OUT := IN1 modulo IN2
25	1	SHL	OUT := IN1 « IN2
	2	SHR	OUT := IN1 » IN2
	3	ROR	OUT := IN1 rotate right IN2
	4	ROL	OUT := IN1 rotate left IN2
26	5	AND	OUT := IN1 & IN2
	6	OR	OUT := IN1 or IN2
	7	XOR	OUT := IN1 xor IN2
	8	NOT	OUT := not IN1
27	2a	MAX	Compares the value in the accu and operand and provides the larger of the two
	2b	MIN	Compares the value in the accu and operand and provides the smaller of the two
28	5	GT	OUT := IN1 > IN2
	6	GE	OUT := IN1 >= IN2
	7	EQ	OUT := IN1 = IN2
	8	LE	OUT := IN1 <= IN2
	9	LT	OUT := IN1 < IN2
	10	NE	OUT := IN1 <> IN2
30	1a	ADD	TIME := TIME + TIME
	3a	ADD	DT := DT + TIME
	4a	SUB	TIME := TIME - TIME
	8a	SUB	DT := DT - TIME
	9a	SUB	TIME := DT - DT

9.2.2 IL syntax elements

No. of the table	Attribute	Description	Example/explanation
52 (1/3)	1	LD	Synchronises the current result with the operand LD %IR0 LDN %MX4.0
	2	ST	Saves the current result to the operand address ST %QR0 STN %MX1000.2
	4	AND	Logical AND AND 8 ANDN 12
	6	OR	Logical OR OR 8 ORN 12
	7	XOR	Logical exclusive OR XOR 8 XORN 12
	7a	NOT	Logical negation (of a complement) NOT %MX0:1
	8	ADD	Addition LD 3 ADD 4 LD T#12s ADD T#1h23m LD DT#1984-06-25-15:36:55 ADD T#1h23m

No. of the table	Attribute	Description	Example/explanation
52 (2/3)	9	SUB	Subtraction LD 3 SUB 2 LD T#1d3h12s SUB T#1h23m LD DT#1984-06-25-15:36:55 SUB T#1h23m LD DT#1988-06-25-15:36:55 SUB DT#1984-06-25-15:36:55
	10	MUL	Multiplication LD 2 MUL 3
	11	DIV	Division LD 12 DIV 6
	11a	MOD	Modulo division LD 12 MOD 5
	12	GT	Comparison: › LD 12 GT 10
	13	GE	Comparison: ›= LD 12 GE 10
	14	EQ	Comparison: = LD 12 EQ 12
	15	NE	Comparison: ‹› LD 12 NE 10

No. of the table	Attribute	Description	Example/explanation
52 (3/3)	16	LE	Comparison: <= LD 12 LE 13
	17	LT	Comparison: < LD 12 LT 10
	18	JMP	Jump to the point JMP {label} JMPC {label} conditional JMPCN {label} conditional not LD %IDo LT 10 JMPCN label1 ADD 3 label1: SUB 2

9.3 Instruction list error codes

The instruction list script is run through line-by-line at the time of every measurement once all of the measurement values have been generated. If an error occurs during this process, the script execution stops and enters the relevant error code in the device log. The number of the line in which the error occurred is entered as the error code parameter. Instructions on evaluating the device log are included in the chapter "Evaluating the device log" (see "Evaluating the device log" on page 136).

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3001	UNKNOWN ERROR	## [1)]	---	Unknown error
3002	DECODING ERROR	## [1)]	---	Error when decoding the script line
3003	OPERATOR ERROR	## [1)]	---	Invalid operator (see "IL syntax elements" on page 74 and "Expansions" on page 78)
3004	TYPE MISMATCH	## [1)]	---	Data type of the operand and accu are not compatible.
3005	PARSE ERROR	## [1)]	---	No label, comment, operator or operand found in the script line

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3006	UNKNOWN OPERAND	## [1]	---	Invalid operand (see "Common elements" on page 72, table no. 4, no. 7, no. 8, no. 15)
3007	INVALID DIRECT ADDRESS	## [1]	---	No valid address
3008	INVALID NUMERIC OPERAND	## [1]	---	Invalid numeric operand
3009	UNKNOWN LABEL	## [1]	---	The label to which the jump is to be made does not exist.
3010	INVALID LOCATION	## [1]	---	Invalid specification of the storage location (see "Common elements" on page 72 table no. 15)
3011	SIZE NOT SUPPORTED	## [1]	---	Invalid specification of the type when reading or writing to the storage location (see "Common elements" on page 72 and "Expansions" on page 78 table no. 15 respectively)
3012	NOT ABLE TO WRITE TO OPERAND	## [1]	---	An attempt is being made to write to a read-only operand.
3013	MAX NUMBER OF LABELS REACHED	## [1]	---	The number of defined labels exceeds the maximum permissible number for this device class.
3014	LABEL ALREADY DEFINED	## [1]	---	A label with exactly the same title has already been defined in previous script lines.
3015	MAX INSTRUCTIONS PER RUN EXCEEDED	## [1]	---	The number of script commands exceeds the maximum permissible number for this device class.
3016	UNKNOWN TYPE	## [1]	---	Unknown data type of the operand
3017	INVALID TIME OPERAND	## [1]	---	Invalid specification of a "duration" type operand
3018	INVALID DATE AND TIME OPERAND	## [1]	---	Invalid specification of a "date and time" type operand

[1] Number of the script line in which the error has occurred

9.4 Expansions

The standard version was extended with the following expansions:

No. of the table	Description	Example/explanation
15	R (real number, (32 bits) size)	%IR0 %QR2 %MR1000
	T (duration - size)	LD T#1d23h45m ST %MT0
	S (date and time - size)	LD DT#2010-08-25-15:36:55 ST %MS0
52 (1/2)	DOW Day of week	Determine the day of the week from a "date and time" type of variable (* 0 = Monday *) DOW DT#2010-08-25-15:36:55
	LDS [1)] Load status	(* Load status from %IR1*) LDS %IR1
	STS [1)] Store status	(* Set %QR1 status to 2 *) LD 2 STS %QR1
	POW Power of	The variable can be exponentiated with the operator "POW". LD %IR0 POW 2.0

No. of the table	Description	Example/explanation
52 (2/2)	INC Increment	Increases a variable LD 5 INC
	DEC Decrement	Reduces a variable LD 5 DEC
	JGTZ	Jump to the label, if the value in the accu › 0 JGTZ {label} LD 5 JGTZ label1 ADD 3 label1: SUB 2
	JGEZ	Jump to the label, if the value in the accu >= 0
	JZ	Jump to the label, if the value in the accu = 0
	JNZ	Jump to the label, if the value in the accu ‹› 0
	JLEZ	Jump to the label, if the value in the accu <= 0
	JLTZ	Jump to the label, if the value in the accu ‹ 0
	CEIL	Round up the value in the accu
	FLOOR	Round down the value in the accu
	ROUND	Commercially round the value in the accu

[1)]rapidM2M developer package M11 uses a special coding to indicate various error statuses in the measurement channels (e.g. OL = open loop, SC = short circuit). This error code that is encoded in the normal measurement value must not be used for calculations. This error code can be read or written with both "LDS" and "STS" operators. The "LDS" operator always provides a result from the "UINT" type. The "STS" operator expects a parameter from the "UINT" type.

Possible error codes

Status	Description
0	Value valid
1	Undefined error
2	Overflow
3	Underflow
4	Open loop (interruption)
5	Short circuit

9.5 Device-specific implementation

Maximum permissible number of script commands	100
Maximum number of characters for a syntax element	25
Maximum number of characters for a script line	100
Maximum permissible number of labels	20
Input addresses	%Ix0 .. %Ix64 • %Ix0: Internal measuring channel "GSM level" • %Ix1 to %Ix64 or %Ix1000 to %Ix1063: Interface channels 1-64
Output addresses	%Qx1000 .. %Qx1063 • %Qx1000 to %Qx1063: Output channels 1-64
Memory register addresses	%Mx0 .. %Mx31

Global register addresses	%Mx1000 .. %Mx1001
	• %Mx1000: Global trigger mask
	  • Bit0: Warning
	  • Bit1: Alarm
	  • Bit2: Fault warning
	  • Bit3: Fault alarm
	  • Bit4-7: Reserved
	  • Bit8: Quick recording
	  • Bit9: Slow recording
	  • Bit10: Reserved
	  • Bit11: Switch on recording
	  • Bit12: Switch off recording
	  • Bit13: Initiate transmission
	  • Bit14: 0...Trigger is initiated when "greater than or equal" to the level, 1...Trigger is initiated when "lower than or equal" to the level
	  • Bit15: Activate online mode
	  • Bit16: Warm-up time active
	  • Bit17: Measurement active
	  • Bit18-19: Reserved
	• %Mx1001: Current time stamp

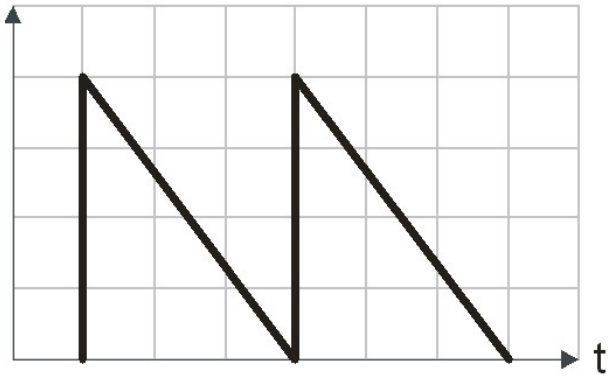
Register addresses that specify which triggers are active or inactive for the relevant input.	%Mx2000 .. %Mx2064 • Bit0-7: Reserved • Bit8: Quick recording • Bit9: Slow recording • Bit10: Reserved • Bit11: Switch on recording • Bit12: Switch off recording • Bit13: Initiate transmission • Bit14: 0...Trigger is initiated when "greater than or equal" to the level, 1...Trigger is initiated when "lower than or equal" to the level • Bit15: Activate online mode
Register addresses that specify which alarms are active or inactive for the relevant input.	%Mx3000 .. %Mx3064 • Bit0: Warning • Bit1: Alarm • Bit2: Fault warning • Bit3: Fault alarm
Display register addresses	%Mx4000 .. %Mx4007

9.6 Example

9.6.1 Saw-tooth generator

This example generates a saw-tooth on the output of a device.

```
LD %MR0          (* Loads the value from the memory register 0 *)
LE 0.0           (* Checks <= 0.0 *)
JMPCN end        (* If the test of the previous line does not equate to 0,
                    jump to the "END" label *)
LD 4.0           (* Load value 4 into the accu *)
ST %MR0          (* Save value 4 from the accu to memory register 0 *)
end:             (* "END" label *)
LD %MR0          (* Loads the value of memory register 0 *)
SUB 0.1          (* Subtract 0.1 from the value in the accu *)
ST %QR0          (* Write the value from the accu to output 0 *)
ST %MR0          (* Write the value from the accu to memory register 0 *)
```

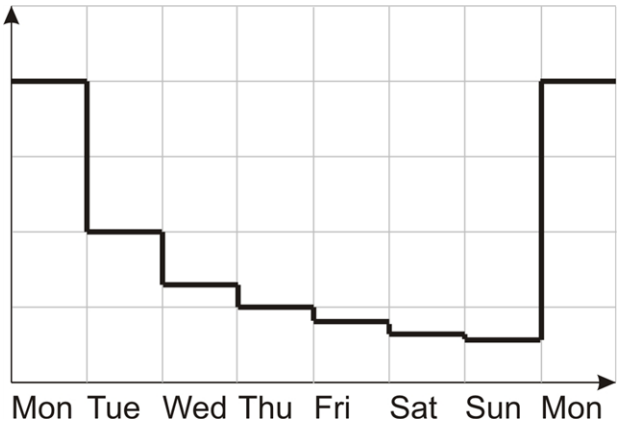


Saw-tooth generated by means of IL

9.6.2 Weekday designation

This example determines the weekday from the current time stamp of a device. An interim value is then calculated by dividing the time difference between the cycles of the script by six. With a measurement cycle of 2 minutes this equates to a value of 20. This interim value is then also divided by the determined weekday and the result is issued at output 0.

```
LD %MS0                 (* Loads the time stamp from memory register 0 *)
EQ  DT#0-00-0-0:0:0     (* Checks = 0-00-0-0:0:0 (No time stamp available yet) *)
JMPC END                (* If the test of the previous line equates to 1, jump to
                           the "END" label *)
DOW %MS1001             (* Determine weekday from the current time stamp *)
ADD 1                   (* +1 so that when the division is completed, it is not divided
                           by 0 (Monday) *)
ST %MD2                 (* Save the determined weekday in memory register 2)
LD %MS1001              (* Load current time stamp into the accu *)
SUB %MS0                (* Subtract the time stamp that is saved in memory register
                           0 from the accu *)
DIV T#6s                (* Divide by 6 *)
ST %MT1                 (* Convert the "duration" to a "real number" by
LD %MR1                    saving it in and loading it from memory register 1 *)
DIV %MR2                (* Divide by the weekday *)
ST %QR0                 (* Write the value from the accu to output 0 *)
END:                    (* "END" label *)
LD %MS1001              (* Load current time stamp *)
ST %MS0                 (* Write current time stamp to memory register 0 *)
```



Output signal generated by means of IL

6872
6872
6872

Chapter 10 Pawn script

10.1 General

The following chapter describes the functionality of the pawn script.

Additional information is provided on the homepage of the developers:
http://www.compuphase.com/pawn/pawn.htm

There are two ways to create a pawn script for the rapidM2M developer package M11 :

- Direct entry in the "Script" input field in the "Control" configuration section
- Use of a script template, that was created on the myDataweb server

10.1.1 Direct input of a pawn script

The pawn script is entered via the "Control" configuration section (see "Control" on page 55) of the input screen for configuring the site. "Pawn" must be selected as the "script type" so that rapidM2M developer package M11 interprets the commands entered in the "script" as a pawn script.

10.1.2 Using a script template

Predefined script templates can be selected via the "Script template" list selection in the "Control" configuration section (see "Control" on page 55) of the input screen for configuring the site (to create a template, see "myDataweb-Server user manual " 206.886). After selecting the control program, the template is copied to the "Script" input window where it can be adapted. When using this method, "Pawn" must also be selected as the "script type" so that the rapidM2M developer package M11 interprets the commands as a pawn script.

> ***Note:*** *If a pawn script is used to write directly to an output, the setpoint entered via the input screen of the myDataweb server is overwritten with the calculated value.*

10.2 General syntax

10.2.1 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.

10.2.2 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.

10.2.3 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.

10.2.4 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.

10.2.5 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			

10.2.6 Numerical constants

10.2.6.1 Numerical integer constants

Binary:

ob followed by a series of 0 and 1

Decimal:
A series of numbers between 0 and 9

Hexadecimal:
0x followed by a series of numbers between 0 and 9 and the letters a to f

10.2.6.2 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.

10.3 Variables

10.3.1 Declaration

The keyword "new" declares a new variable. For special declarations, the keyword "new" is replaced with "static" (see "Static local declaration" on page 88). 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 99) - local variable

Example:

```
new a;     // without initialisation (value is 0)
new b = 3; // with initialisation (value is 3)
```

10.3.2 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.

10.3.3 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.

10.3.4 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.

10.3.5 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.

10.3.6 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)
```

10.4 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 104.

10.5 Array variables

10.5.1 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]".

10.5.2 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 */
```

10.5.3 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
```

10.5.4 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 90).

10.5.5 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.

10.6 Operators and expressions

10.6.1 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*

10.6.2 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
```

10.6.3 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".*

10.6.4 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.

10.6.5 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

10.6.6 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.

10.6.7 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".

10.6.8 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 o 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.

10.6.9 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

10.7 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.

10.7.1 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.

10.7.2 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".

10.7.3 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.

10.7.4 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.

10.7.5 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
}
```

10.7.6 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
}
```

10.7.7 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
}
```

10.7.8 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
}
```

10.7.9 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.

10.7.10 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.

10.7.11 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).

10.7.12 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
}
```

10.7.13 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.

10.7.14 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
}
```

10.7.15 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
}
```

10.8 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 91.

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.

10.8.1 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 */
```

10.8.2 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.

10.8.3 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]
}
```

10.9 Extensions

10.9.1 Constants

Structure of a TablePoint entry

```
Mdn_TablePoint[Float:.key, Float:.value]
```

Returncodes for general purposes

```
OK    = 0
ERROR = -1
```

Error codes of the Mdn_CalcTable function

```
MDN_TAB_ERR_FLOOR = -1
MDN_TAB_ERR_CEIL  = -2
```

Status of the measurement value channel

```
const Mdn_ValueStatus:
{
  MDN_STATUS_OK  = 0,     // Value is OK
  MDN_STATUS_NAN = 1,     // Value is NAN
  MDN_STATUS_OF  = 2,     // Channel is overflow
  MDN_STATUS_UF  = 3,     // Channel is underflow
  MDN_STATUS_OL  = 4,     /* Cable break was detected or no sensor
                             was connected */
  MDN_STATUS_SC  = 5,     // Short circuit was detected
};
```

Alarm flags

```
const
{
  MDN_FLG_WARNING   = 0b00000001,  // Warning
  MDN_FLG_ALARM     = 0b00000010,  // Alarm
  MDN_FLG_FAULTLOW  = 0b00000100,  // Technical error of low priority
  MDN_FLG_FAULTHIGH = 0b00001000,  // Technical error of high priority
  MDN_FLG_UNDERFLOW = 0b10000000,  // Shortfall of the alarm level
};                                 /* Must be set if the value is
                                      below the threshold.
                                      If this flag is not set, it
                                      is assumed that the threshold has been
                                      exceeded */
```

Trigger flags

```
const
{
  // Device-specific
  MDN_TRG_INTERN1    = 0b0000000000010000,
  // Device-specific
  MDN_TRG_INTERN2    = 0b0000000000100000,
  // Device-specific
  MDN_TRG_INTERN3    = 0b0000000001000000,
  // Device-specific
  MDN_TRG_INTERN4    = 0b0000000010000000,
  // Request fast recording
  MDN_TRG_RECQUICK   = 0b0000000100000000,
  // Request slow recording
  MDN_TRG_RECSLOW    = 0b0000001000000000,
  // Start recording the current measurement
  MDN_TRG_RECORD_ON  = 0b0000100000000000,
  // Avoid recording the current measurement
  MDN_TRG_RECORD_OFF = 0b0001000000000000,
  // Request an individual connection with the server
  MDN_TRG_TXSTART    = 0b0010000000000000,
  // Request an online connection with the server
  MDN_TRG_CONTINOUS  = 0b1000000000000000,
};
```

10.9.2 Device-dependent channel numbers

Definition of the constants	Explanation
MDN_CH_GSM	Internal measurement channel "GSM level"
MDN_CH_AIR232_IN1 : MDN_CH_AIR232_IN64	Interface channels 1-64
MDN_CH_AIR232_OUT1 : MDN_CH_AIR232_OUT64	Interface output channels 1-64
MDN_NUM_CHANNELS	Total number of all channels (all of the in- and outputs)

10.9.3 Functions

10.9.3.1 Core functions

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.*

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*

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.*

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.*

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 the value of an argument from a variable argument list. If the argument is an array, the "index" specifies the index of the required array element.*

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.*

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.*

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)*

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	*Smaller of the two transferred values*

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*

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"*

10.9.3.2 Console functions

print(const string[] value);

Prints the specified value on the standard output

Parameter	*Explanation*
value	*The character string to be issued*

printi(value);

Prints the specified value on the standard output

Parameter	*Explanation*
value	*The integer type value to be issued*

printf(Float:value);

Prints the specified value on the standard output

Parameter	*Explanation*
value	*The float type value to be issued*

10.9.3.3 Mdn functions

getapilevel();

Issues the implemented API level of the script engine

Mdn_GetCh(ch, &Float:value, &Mdn_ValueStatus:status = MDN_STATUS_OK);

Reads out the current value and status of a channel

Parameter	*Explanation*
ch	*Number of the channel (see "Device-dependent channel numbers" on page 109)*
value	*Value of the channel*
status	*Status of the channel (see "Status of the measurement value channel" in the chapter "Constants" on page 108) - OPTIONAL*

Mdn_SetCh(ch, Float:value, Mdn_ValueStatus:status = MDN_STATUS_OK);

Sets the value and status of a channel

Parameter	*Explanation*
ch	*Number of the channel (see "Device-dependent channel numbers" on page 109)*
value	*Value of the channel*
status	*Status of the channel (see "Status of the measurement value channel" in the chapter "Constants" on page 108) - OPTIONAL*

Mdn_GetTime(&hour=0, &minute=0, &second=0);

Reads the current time and saves it in three separate fields

Parameter	*Explanation*
hour	*Hour of the current time - OPTIONAL*
minute	*Minute of the current time - OPTIONAL*
second	*Second of the current time - OPTIONAL*

	Explanation
Return value	*Seconds since 01/01/2000*

Mdn_GetDate(&year=0, &month=0, &day=0);

Reads the current date and saves it in three separate fields

Parameter	*Explanation*
year	*Year of the current time - OPTIONAL*
month	*Month of the current time - OPTIONAL*
day	*Day of the current time - OPTIONAL*

	Explanation
Return value	*Seconds since 01/01/2000*

Mdn_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*

Mdn_GetAlarm(ch);

Returns the alarm status of a channel

Parameter	*Explanation*
ch	*Number of the channel (see "Device-dependent channel numbers" on page 109)*

	Explanation
Return value	*Positive, if successful (see "Alarm flags" in the chapter "Constants" on page 108), ERROR, if no valid channel is specified.*

Mdn_SetAlarm(ch, alarm, Float:value, Float:level = 0.0);

Sets the alarm status of a channel

Parameter	*Explanation*
ch	*Number of the channel (see "Device-dependent channel numbers" on page 109)*
value	*Value of the channel used for the alarm calculation*
level	*Includes the alarm threshold - OPTIONAL*

Mdn_SetTrigger(trigger);

Sets global trigger events

Parameter	*Explanation*
trigger	*Trigger flags (see "Trigger flags" in the chapter "Constants" on page 108).*

	Explanation
Return value	*OK, if successful*

Mdn_CalcTable(Float:key, &Float:value, const table[][Mdn_TablePoint], size = sizeof table);

Searches for a certain key in the supplied table and returns the relevant value. The value is interpolated between the two points, if the key is between two values in the table.

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 Mdn_TablePoint type table.*
size	*Number of elements in the table*

	Explanation
Return value	• *OK, if the relevant value was found* • *MDN_TAB_ERR_FLOOR, if the key is smaller than the first table entry. "value" comprises the first table entry.* • *MDN_TAB_ERR_FLOOR, if the key is larger than the last table entry. "value" comprises the last table entry.*

10.9.3.4 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)

fround(Float:x);

Commercially rounds the transferred float

Parameter	*Explanation*
x	*Float that is to 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:

Float:sin(Float:x);

Sine of x

Float:cos(Float:x);

Cosine of x

Float:tan(Float:x);

Tangent of x

Float:asin(Float:x);

Arcsine(x) in the range [-π/2, π/2], x element of [-1, 1]

Float:acos(Float:x);

Arccosine(x) in the range [0, π], x element of [-1, 1]

Float:atan(Float:x);

Arctangent(x) in the range [-π/2, π/2]

Float:atan2(Float:y, Float:x);

Arctangent(y/x) in the range [-π, π]

Float:sinh(Float:x);

Hyperbolic sine of x

Float:cosh(Float:x);

Hyperbolic cosine of x

Float:tanh(Float:x);

Hyperbolic tangent of x

Float:exp(Float:x);

Exponential function e^x

Float:log(Float:x);

Natural logarithm ln(x), x › 0

Float:log10(Float:x);

Logarithm as the basis 10 $\log_{10}(x)$, x › 0

Float:pow(Float:x, Float:y);

x^y. An argument error has occurred if x = 0 and y ‹= 0, or if x ‹ 0 and y is not a whole number.

Float:sqrt(Float:x);
Root x, x >= 0

Float:ceil(Float:x);
Smallest whole number that is not smaller than x

Float:floor(Float:x);
Largest whole number that is not larger than x

Float:fabs(Float:x);
Absolute value | x |

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

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.

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.

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.

10.9.4 Pawn script error codes

The PAWN script is run through at the time of every measurement once all of the measurement values have been generated. If an error occurs during this process, the script execution is stopped and deactivated. The relevant error code is also entered in the device log. Instructions on evaluating the device log are included in the chapter "Evaluating the device log" (see "Evaluating the device log" on page 136).

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3001	AMX_ERR_EXIT	---	---	Abortion
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
3006	AMX_ERR_INVINSTR	---	---	Invalid statement
3007	AMX_ERR_ STACKLOW	---	---	Stack underflow

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3008	AMX_ERR_ HEAPLOW	---	---	Heap underflow
3009	AMX_ERR_ CALLBACK	---	---	No (valid) 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 installed (or installed twice)
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.

10.10 Example

10.10.1 Saw-tooth generator

This example generates a saw-tooth on the output of a device.

```
#include <mdlAIR232>                    /* Load the definitions for the
                                        rapidM2M developer package M11
                                                */

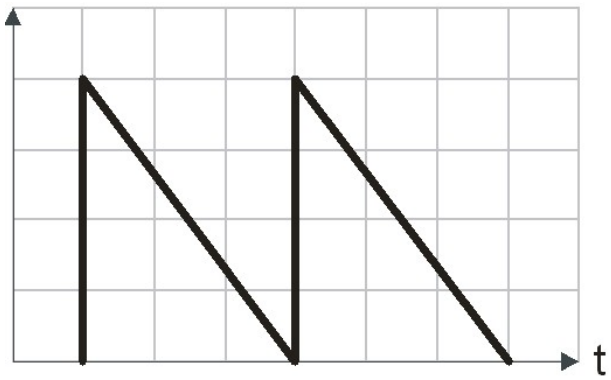
forward public Mdn_CtrlFinish();        /* Declaration of the function that is called up at the end of the
                                           measurement cycle. */

static Float:fCurrentValue;             // Static declaration of the fCurrentValue variable of the float type
                                        // This variable is not reset between two measurement cycles
main()
{
  fCurrentValue = 4.0;                  // Initialise fCurrentValue with the value 4.0
}

public Mdn_CtrlFinish()
{
  if(fCurrentValue <= 0.0)              // If the variable is less or equal to 0.0,
     fCurrentValue = 4.0;               // The variable is set to 4.0

  fCurrentValue -= 0.1;                 // Subtract 0.1 from the variable

  Mdn_SetCh(MDN_CH_IOUT1, fCurrentValue); // Set output 1 to the value of the fCurrentValue variable
}
```



Saw-tooth generated by means of pawn

10.10.2 Weekday designation

This example determines the weekday from the current time stamp of a device. An interim value is then calculated by dividing the time difference between the cycles of the script by six. With a measurement cycle of 2 minutes this equates to a value of 20. This interim value is then also divided by the determined weekday and the result is issued at output 0.

```
#include <mdlAIR232>                                  /* Load the definitions for the
                                                      rapidM2M developer package M11
                                                                  */

forward public Mdn_CtrlFinish();                      /* Declaration of the function that is called up at the
                                                         end of the measurement cycle. */

static iLastStamp;                                    /* Static declaration of the
                                                         iLastStamp variable of the integer type
                                                         This variable is not reset between
                                                         2 measurement cycles*/

main()
{
  iLastStamp = 0;                                     // Initialise iLastStamp with the value 0
}

public Mdn_CtrlFinish()
{
  new iDayOfWeek;                                     /* Declaration of the iDayOfWeek variable of the
  new iCurrentStamp;                                     integer type */
  new Float:fResult;                                  /* Declaration of the iCurrentStamp variable of the
                                                         integer type */
                                                      /* Declaration of the fResult variable of the
                                                         float type
                                                         These variables are reset with every
                                                         measurement cycle */

  if(iLastStamp != 0)                                 /*  The content of the brackets is executed,
  {                                                      if the content of the iLastStamp variable
                                                         has already been set. */

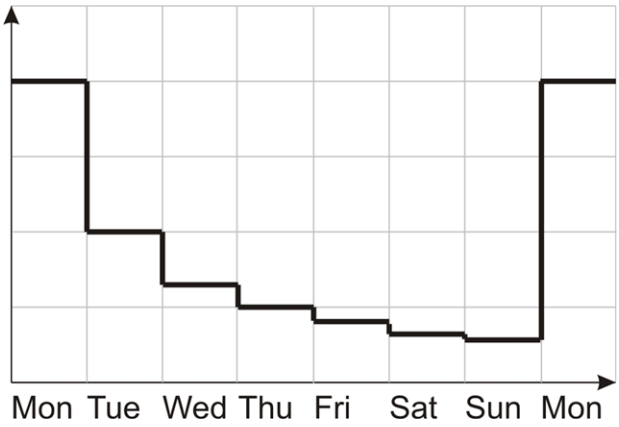
    iDayOfWeek = Mdn_DoW(Mdn_GetDate());              /* Save weekday to the iDayOfWeek
                                                         variable */
    iDayOfWeek += 1;                                  /* +1 so that the subsequent division is not
                                                         divided by 0 (Monday) */

    iCurrentStamp = Mdn_GetTime();                    /* Read out the current time and save to the
                                                         iCurrentStamp variable */

    fResult = (iCurrentStamp - iLastStamp) / 6.0;     /* Determine the time difference between the current and last
                                                         time of a measurement (measurement cycle) and
                                                         divide by 6 */
    fResult = fResult / iDayOfWeek;                   // Divide the result by the weekday

    Mdn_SetCh(MDN_CH_IOUT1, fResult);                 // Write the result to the output
  }

  iLastStamp = Mdn_GetTime();                         /* Temporarily save the current time for processing
}                                                        until the next time of a measurement */
```



Output signal generated by means of pawn

10.10.3 Calculating the flow rate with the table

This example determines the flow rate of a sewer based on the fill level of the sewer. The fill level is measured at input 1 and the flow rate should be issued at output 1.

```
#include <mdlAIR232>                                  /* Load the definitions for the
                                                      rapidM2M developer package M11
                                                                */


forward public Mdn_CtrlFinish();                      /* Declaration of the function that is called up at the
                                                         end of the measurement cycle. */
static const Table[][Mdn_TablePoint] =                /* Static table for converting the height
[                                                        to the flow rate */
  [ 0.0,  0.0],
  [ 2.0,  1.0],
  [ 5.0,  3.0],
  [10.0, 10.0],
  [50.0, 15.0]
];

main()
{
}

public Mdn_CtrlFinish()
{
  new Float:fHeight;                                  // Declaration of the fHeight variable of the float type
  new Float:fFlow;                                    // Declaration of the fFlow variable of the float type
  new Mdn_ValueStatus:iStatus;                        /* Declaration of the iStatus variable of the
                                                         Mdn_ValueStatus type
                                                         These variables are reset
                                                         with every measurement cycle */


  Mdn_GetCh(MDN_CH_UI1, fHeight, iStatus);            /* Read out channel 1 and save the value to
                                                         the fHeight variable and the status to
                                                         the iStatus variable */


  if(iStatus == MDN_STATUS_OK)                        /* If the status of channel 1 is OK, execute the
  {                                                      following code in the {} brackets */
    new iResult;                                      // Declaration of the iResult variable of the integer type

    iResult = Mdn_CalcTable( fHeight, fFlow,          /* This function calculates the flow rate
                             Table, sizeof Table);       (fFlow) based on the height (fHeight)
                                                         and the table (Table).
                                                         The result is linearly interpolated,
                                                         if the height is between 2 entries.
                                                         The return value comprises the error value of
                                                         the calculation */

    switch(iResult)
    {
      case MDN_TAB_ERR_FLOOR:                         /* If the height is smaller than the first entry in the
      {                                                  table */
        fFlow = 0.0;                                  // Set the flow rate to 0.0
      }
      case MDN_TAB_ERR_CEIL:                          /* If the height is larger than the last entry in the
      {                                                  table */
        fFlow = 0.0;                                  // Set the flow rate to 0.0
      }
    }
  }

  Mdn_SetCh(MDN_CH_IOUT1, fFlow);                     //Write the flow rate to output 1
}
```

10.11 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 86).
- 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 104). The "placeholder" argument replaces some applications of the ZERO pointer (see "Standard values of function arguments" on page 106).
- Number can be specified in a hexadecimal, decimal or binary format. The octal format is not supported (see "Numerical constants" on page 86). 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 101). 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 95.
- An empty statement is an empty block (with {}) and not a semicolon (see "Composite statements" on page 97). 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. For positive denominators, this means that the truncation direction for the operator "/" is always towards the smaller value (whereby -2 is less than -1).

The "%" operator always generates a positive result regardless of the prefix of the counter (see "Operators and expressions" on page 91).

- There is no unary "+" operator that in any event is a "no-operation" operator ("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 declaration of functions (i.e. prototypes) is not necessary. A pawn is a two-pass compiler. It detects all of the function 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.

Chapter 11 Troubleshooting

11.1 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; no output of the debug information on COM2)	• Check the cable connections (see "Connection of the sensor and power supply" on page 39) • 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 M11 module (J104) " on page 28). • Check that the rapidM2M module M11 is correctly inserted in the holding fixture on the rapidM2M Developer Board (see "Inserting the M11 module" on page 38).
Communication problems	• Load the device log from the myDataweb server and use myDatanetDeviceConfig for the evaluation (see "Evaluating the device log" on page 136).
Not all or no data is available on the server	• The connection was interrupted during transmission, which is indicated by a time-out entry in the connection list (see "myDataweb-Server user manual " 206.886). Solution: Initiate a ALOHA transmission or wait for the next cyclical transfer. • The assignment of the device and site is not correct (see "Site" on page 55).
The interface channel data is not plausible	• Check that the column selection is set correctly in the interface channel configuration (see "Interface channels 1-32" on page 56). • Check the configuration of the serial interface (see "Serial interface" on page 60). The number format or the number delimiter(s) may be configured incorrectly. • Check that the jumper for supplying the level shifters (J111) is closed (see "Jumper for supplying the level shifters (J111)" on page 24).

Problem	Cause/solution
No output of the debug information on COM1	• Check the cable connections (see "Connection of the sensor and power supply" on page 39) • Check whether the configuration of your terminal program corresponds to the settings specified in chapter "RS232 interface COM1 (J400)" on page 23. • Check that the jumper for supplying the level shifters (J111) is closed (see "Jumper for supplying the level shifters (J111)" on page 24). • If there are less columns in the data message than specified during the configuration of the interface channels (see "Interface channels 1-32" on page 56), this is recognised as an invalid frame, no data is recorded and a corresponding entry is generated in the device log (see "Error codes of the UART2 interface " on page 51).
Alarm state of an interface channel was not identified	Increase the measurement cycle
Alarm state was not transmitted although the data is present	• Check the alarm settings of the measurement channel • The connection was interrupted during transmission, which is indicated by a time-out entry in the connection list (see "myDataweb-Server user manual " 206.886). Solution: Initiate a ALOHA transmission or wait for the next cyclical transfer.
Alarm message was not sent although the alarm is signalled	• Check the settings of the alarm call plan (see "myDataweb-Server user manual " 206.886). • Check the address data of the alarm call plan (see "myDataweb-Server user manual " 206.886).
The value set via the input screen on the myDataweb server is not issued at the output channel.	• The site settings were overwritten by a server-PLC. • Check whether the output channel is being written to via the script. The value specified via the input screen on the myDataweb server is overwritten with a value calculated by the script.
The instruction list is not being executed correctly.	• Check that the correct script type was selected during the configuration of the control (see "Control" on page 55). • Load the device log from the myDataweb server and use myDatanetDeviceConfig for the evaluation (see "Evaluating the device log" on page 136). A list of all the possible instruction list error codes is included in the chapter "Instruction list error codes" (see "Instruction list error codes" on page 76).

Problem	Cause/solution
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 55). • Load the device log from the myDatawebserver and use myDatanetDeviceConfig for the evaluation (see "Evaluating the device log" on page 136). 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 119).

11.2 Log entries and error codes

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1020	ERROR SOD	##	---	Internal system error There may be a hardware problem if this error is included in the device log several times with the same parameter code. Contact the manufacturer in this case (see "Contact information" on page 145).
1200	MODEM ERROR			Modem error (see "Modem error" on page 133)
1212	ERROR MODEM IRREGULAR OFF	##	---	Indicates a faulty connection. The parameter indicates how many consecutive connections have not worked.
1252	MODEM TO CON	##	---	Timeout while a connection is being established. The parameter includes 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 the error several times (see "Contact information" on page 145).
1282	ZLIB STREAMFINISH ERR	##	---	Internal error Contact the manufacturer if the device log includes the error several times (see "Contact information" on page 145).
1440	DECAY MEM ERR	---	---	reserved for extensions

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1600 (1/2)	STATE (1/2)	1	OK	Last connection OK
		2	TRANSMISSION ERROR	Last transmission faulty Try again later
		3	MEASUREMENT ERROR	Last measurement faulty • Check the configuration • Check the connection between the device and the sensor
		4	HOLD	Standby (GPRS ON, measurement OFF) If the ALOHA transmission mode is initiated, the rapidM2M developer package M11 switches back into "RUN" mode (GPRS ON, measurement ON).
		5	TRANSPORT	Transport lock (GPRS OFF, measurement OFF) see "Standby"
		6	OFFLINE	Offline (GPRS OFF, measurement ON) see "Standby"
		7	NO NET (NET LOCK)	Netlock/no matching provider [1)] • Try to improve the position of the antenna. • Check whether the device is in the coverage area (www.microtronics.at/footprint). • Lift the network block (see "myDatanetDeviceConfig user manual " 206.887)
		8	NO NET	No GSM network • Improve the position of the antenna • Try again later
		10	GPRS ERROR	No GPRS connection Try to improve the position of the antenna.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1600 (1/2)	STATE (1/2)	11	IP ERROR	No myDataweb server available • Check whether port 51241 is enabled on the myDataweb server. • Try again later
		12	SIM CARD ERROR	Faulty SIM chip Contact support.
2000	MODULE ERR			Module-specific error (see "Error codes of the UART2 interface " on page 51).
3000 - 3099				Error codes of the script execution. The following error descriptions apply depending on the selected script type: • "Instruction list error codes" on page 76 • "Pawn script error codes" on page 119

[1] A netlock can only be activated on devices with a USB connector in conjunction with the myDatanetDeviceConfig configuration program.

11.2.1 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 the settings for the netlock or remove the netlock[1)]. • Check whether the device is in the coverage area (www.microtronics.at/footprint).

[1)] A netlock can only be activated on devices with a USB connector in conjunction with the myDatanetDeviceConfig configuration program.

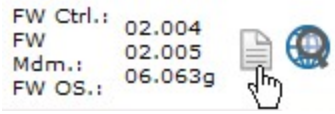
Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1200	BAND SEL FAILED	-969	---	A network could not be found on the GSM900/1800 or on the GSM85071900 band. • Try to improve the position of the antenna. • Check whether the device is in the coverage area (www.microtronics.at/footprint).
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

11.3 Evaluating the device log

11.3.1 Evaluating the device log on the myDataweb server

The last 300 log entries on the myDataweb server can be called up via the button shown below that is located in the measurement device list. As the log entries are sent to the server in the transmission cycle in the same way as the measurement data, only the log entries up to the last server connection are available.



The manual for the server ("myDataweb-Server user manual " 206.886) includes a detailed description of the evaluation of the device log on the myDataweb server.

11.3.2 Evaluating the device log using myDatanetDeviceConfig

A detailed description regarding the evaluation of the device log using myDatanetDeviceConfig is included in the myDatanetDeviceConfig user manual ("myDatanetDeviceConfig user manual " 206.887).

Chapter 12 Spare parts and accessories

12.1 rapid M2M module

Description	Quantity	Order number
rapidM2M module M11	1	300068

12.2 Antennas

Description	Quantity	Order number
Magnetic roof antenna SMA	1	300085
Extension for ANT900 SMA	1	206.807

12.3 Power supply

Description	Quantity	Order number
NG24(U: 24V, I max.: 1A, P max.: 24W)	1	213.814
MDN power device (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 charger/power supply [1)]	1	300028
Snap-on primary plug US for charger/power supply [1)]	1	300029
Snap-on primary plug AUS for charger/power supply [1)]	1	300030

[1)] Not suitable for the plug-in power supply included in the scope of supply of the developer package (300070).

12.4 Adapter

Description	Quantity	Order number
Null modem adapter 9pol. D-Sub female/male	1	206.686

12.5 Other accessories

Description	Quantity	Order number
myDatanet Tool Pen	1	206.646

6872
6872
6872

Chapter 13 Application example

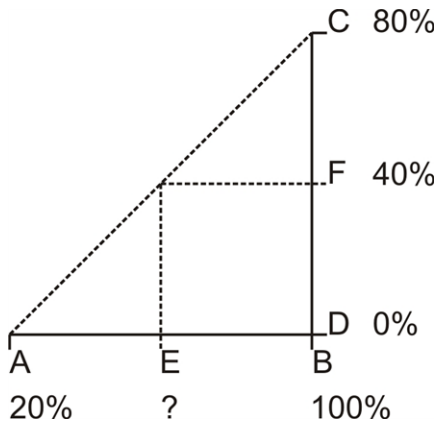
13.1 Lantern control

13.1.1 Specifications

The lantern controller measures the ambient brightness and the power consumption of the lantern. The brightness of the bulb can be adjusted by means of a setpoint for the PWM that is transmitted to the lantern controller to activate the switch. The rapidM2M developer package M11 reads the status from the lantern controller via the RS232 and calculates the setpoint for the PWM via the instruction list, which is then transmitted back to the lantern controller via the RS232 interface. In this simplified example, the brightness of the bulb is only controlled dependent on the ambient brightness. However, a variety of other factors and parameters could be included in the setpoint calculation via the instruction list.

Control specifications:

Ambient brightness [%]	PWM [%]
100	0
80	100
linearly interpolated from 80 to 0	linearly interpolated from 100 to 20
0	20



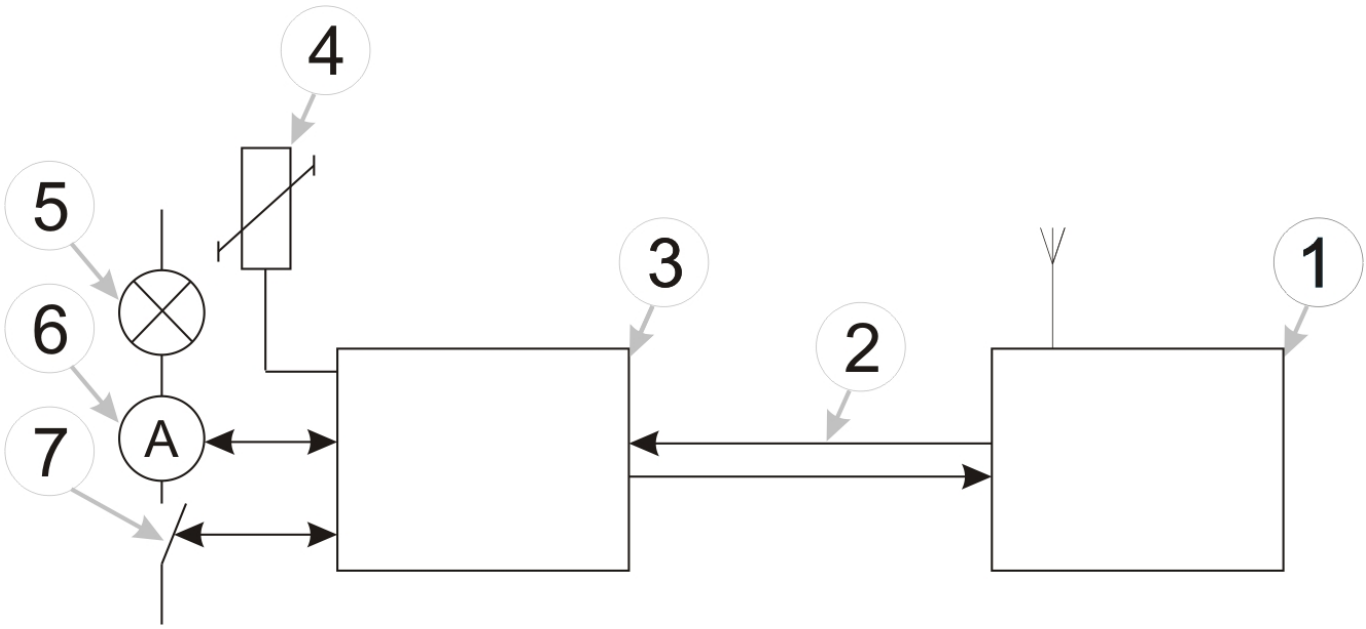
A, B, E = PWM
C, D, F = ambient brightness

Slope = (B-A)/(C-D)
Slope = (100%-20%)/(80%-0%)
Offset = 20%

E = ((B-A)/(C-D)) * F + A
E = ((100%-20%)/(80%-0%)) * 40% + 20% = 60%

Based on:

- Command to initialise the lantern controller following PowerOn: INIT
- Command to read out the status of the lantern controller: RS
- Configuration of the status frame:[STX]‹bulb[ON/OFF]›;‹power consumption [A]›;‹ambient brightness [%]›[ETX]
- Expected control frame: [STX]‹bulb[ON/OFF]›;‹PWM setting [%]›[ETX]



Schematic diagram of the lantern control

1 rapidM2M developer package M11	5 Bulb
2 RS232 interface	6 Current measurement sensor (0-10A)
3 Lantern controller	7 Switch to control the brightness (PWM 0-100%)
4 Ambient brightness sensor (0-100%)	

13.1.2 Required configuration

Only the most important settings to meet the requirements of the described example will be detailed in this section, the complete configuration for the application example will not be provided.

Control	Script type	IL
	Script	see "Instruction list" on page 141

Interface channels1-32 -› Basis	Title 1	"Status of the bulb"
	Interface	RS232
	Column	1
	Title 2	"Power consumption"
	Interface	RS232
	Column	2
	Unit	"A"
	Title 3	"Ambient brightness"
	Interface	RS232
	Column	3
	Unit	"%"
Interface output channels 1-32	Title 1	"Status of the bulb"
	Mode	On
	Title 2	PWM setting
	Mode	On
Serial interface	Data set structure	[STX] to [ETX]
	Number delimiter	;
	Number format	1,000.00
	Init sequence	INIT
	Measurement data sequence	RS

13.1.3 Instruction list

```
LD %IR3      (* Loads the current ambient brightness *)
LE  80.0     (* Checks whether the ambient brightness is <= 80% *)
JMPCN OFF    (* If the test in the previous line does not equal 0, jump to the "OFF" label *)
LD %IR3      (* Loads the current ambient brightness *)
MUL 1.0      (* Multiply with the slope ( (100-20)*(80-0) *)
ADD 20.0     (* Add offset (20) *)
ST %QR1      (* Write calculated value to the "PWM settings" output *)
LD 1         (* Copy value into the accu to activate the bulb *)
ST %QD0      (* Write value to the "Bulb status" output *)
JMP END      (* Jump to the "END" label (skip block to switch off the bulb *)
OFF:         (* "OFF" label *)
LD 0         (* Copy value into the accu to switch off the bulb *)
ST %QD0      (* Write value to the "Bulb status" output *)
END:         (* "END" label *)
```

13.1.4 Communication on the interface

[STX]INIT[ETX]	Command to initiate the lantern controller (Only once during PowerOn)	M11 -› Ctrl
[STX]RS[ETX]	Command to read out the status (during the measurement cycle)	M11 -› Ctrl
[STX]0;0.00;80.0[ETX]	Status: Bulb "0", power consumption "0.00"A, Ambient brightness "0.0"%	Ctrl -› M11 .
[STX]1;100.0[ETX]	Control frame: Bulb "1", PWM setting "100.0"%	M11 -› Ctrl

Chapter 14 Document history

Rev.	Date	Changes
02	05.12.2012	**Chapter "Technical data" on page 11** *Number of available "Interface channels" and "Interface output channels" added* **Chapter "Selection of the GSM network" on page 17** *Added chapter describing the operation of the GSM network selection* **Chapter "Script eradication" on page 19** *Chapter added* **Chapter "Error codes of the UART2 interface " on page 51** *Breakdown of the error codes added* **Chapter "Control" on page 55** *Selection of the pawn script* *Explanation adjusted to the modified input screen* **Chapter "Pawn script" on page 85** *Added chapter describing the functionality of the pawn script* **Chapter "Troubleshooting" on page 127** *Error descriptions and solutions supplemented* **Chapter "Log entries and error codes " on page 130** *Breakdown of the error codes added* **Chapter "Spare parts and accessories" on page 137** *MDN power device (206.623) added*

6872
6872
6872

Chapter 15 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
Tel. +49 (0)6104 971057
Fax. +49 (0)6104 971058
office@microtronics.de
www.microtronics.de

Microtronics-UK Ldt.
9b The Shade, Soham, Ely
Cambs, CB7 5DE
United Kingdom, Europe
Tel. +44 (0)1353 727017
office@microtronics-uk.co.uk
www.microtronics-uk.co.uk