User Manual for the stationary device myDatalogS3



Rev. 02 valid from:

- Firmware version: 01v016
- Modem version: 03v002
- Hardware version: 02v001

CE

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

6872
6872
6872

Chapter 1 Table of contents

User Manual for the stationary device myDatalogS3 1

Chapter 1 Table of contents 3

Chapter 2 Declaration of conformity 11

Chapter 3 Technical data 13

Chapter 4 General specifications 15

4.1 Translation 15

4.2 Copyright 15

4.3 General descriptive names 15

4.4 Safety instructions 15

4.4.1 Use of the hazard warnings 16

4.4.2 Safety and preventative measures for handling GSM/GPRS modems 16

4.4.2.1 Safety and precautionary measures for the GSM/GPRS modem installation 17

4.4.2.2 Safety measures for installing the antenna 17

4.5 Overview 18

4.6 Intended use 19

4.7 General product information 19

4.8 Device labelling 20

4.9 Installation of replacement and wear parts 20

4.10 Storage of the product 20

4.11 Obligation of the operator 21

Chapter 5 Functional Principle 23

5.1 ALOHA transmission mode 23

5.2 Selection of the GSM network 24

5.2.1 Ban list 24

5.3 Script eradication 25

5.3.1 Instruction list 25

5.3.2 Pawn 25

Chapter 6 Storage, delivery and transport 27

6.1 Inspection of incoming deliveries ... 27
6.2 Scope of supply ... 27
6.3 Storage ... 27
6.4 Transportation ... 27
6.5 Return ... 28
Chapter 7 Installation ... 29
7.1 Dimensions ... 29
7.2 Installing the myDatalogS3 ... 29
7.2.1 Top-hat rail assembly ... 30
7.3 Safety instructions for the cabling ... 30
7.3.1 Information on preventing electrostatic discharges (ESD) ... 31
7.4 Electrical installation ... 31
7.4.1 Connecting the sensors, actuators and power supply ... 31
7.4.2 Connecting the GSM antenna ... 35
7.4.3 Technical details about the universal inputs ... 36
7.4.3.1 0/4 to 20mA mode ... 36
7.4.3.2 0 to 2V mode ... 36
7.4.3.3 0 to 10V mode ... 36
7.4.3.4 Standard digital modes (PWM, frequency, digital, day counter, impulse counter) ... 37
7.4.4 Technical details about the outputs ... 37
7.4.4.1 Output 1: Isolated switch contact (NO, CC) ... 37
7.4.4.2 Output 2: Analogue output (mA Out, GND) ... 38
Chapter 8 Initial Start-Up ... 39
8.1 User information ... 39
8.2 General principles ... 39
8.3 Placing the system into operation ... 39
8.4 Testing communication with the device ... 40
Chapter 9 User interfaces ... 43
9.1 User interface on the myDatalogS3 ... 43

9.1.1 Operating elements ... 43
9.1.1.1 Solenoid switch ... 43
9.1.1.2 Status LED ... 44
9.2 User interface on the myDataweb-Server ... 45
9.2.1 Site configuration ... 45
9.2.1.1 Site ... 45
9.2.1.2 Comments ... 45
9.2.1.3 Control ... 46
9.2.1.4 Measurement channels ... 46
9.2.1.4.1 Basis ... 46
9.2.1.4.2 Config ... 49
9.2.1.4.3 Alarms ... 62
9.2.1.4.4 Trigger ... 63
9.2.1.5 Calculated channels ... 63
9.2.1.5.1 Basis ... 64
9.2.1.5.2 Calculation ... 64
9.2.1.5.3 Alarm ... 65
9.2.1.6 Output channels ... 65
9.2.1.6.1 Basis ... 65
9.2.1.6.2 Config ... 67
9.2.1.7 Internal channels ... 67
9.2.1.7.1 Basis ... 67
9.2.1.7.2 Alarms ... 68
9.2.1.7.3 Trigger ... 68
9.2.1.8 Alarm settings ... 69
9.2.1.9 Basic settings ... 70
9.2.1.10 FTP export settings ... 71
9.2.2 Device configuration ... 72
9.2.2.1 Comments ... 72
9.2.2.2 Measurement instrument ... 72

9.2.2.3 Device-specific settings ... 73
9.2.2.4 GPRS ... 74
Chapter 10 Instruction list ... 75
10.1 General ... 75
10.1.1 Direct entry of an instruction list ... 75
10.1.2 Using a script template ... 75
10.2 Implementation details ... 76
10.2.1 Common elements ... 76
10.2.2 IL syntax elements ... 78
10.3 Instruction list error codes ... 80
10.4 Expansions ... 82
10.5 Device-specific implementation ... 84
10.6 Example ... 86
10.6.1 Saw-tooth generator ... 86
10.6.2 Weekday designation ... 86
10.6.3 Using the display registers ... 87
10.6.3.1 Task definition ... 87
10.6.3.2 Required configuration ... 88
10.6.3.3 Instruction list ... 89
10.6.3.4 Communication on the interface ... 90
Chapter 11 Pawn script ... 91
11.1 General ... 91
11.1.1 Direct input of a pawn script ... 91
11.1.2 Using a script template ... 91
11.2 General syntax ... 91
11.2.1 Format ... 91
11.2.2 Optional semicolons ... 91
11.2.3 Comments ... 92
11.2.4 Identifier ... 92
11.2.5 Reserved keywords ... 92

11.2.6 Numerical constants ... 92
11.2.6.1 Numerical integer constants ... 92
11.2.6.2 Numerical floating-point constants ... 93
11.3 Variables ... 93
11.3.1 Declaration ... 93
11.3.2 Local declaration ... 93
11.3.3 Global declaration ... 93
11.3.4 Static local declaration ... 94
11.3.5 Static global declaration ... 94
11.3.6 Floating point values ... 94
11.4 Constant variables ... 94
11.5 Array variables ... 95
11.5.1 One-dimensional arrays ... 95
11.5.2 Initialisation ... 95
11.5.3 Progressive initialisation for arrays ... 95
11.5.4 Multi-dimensional arrays ... 96
11.5.5 Arrays and the "sizeof" operator ... 96
11.6 Operators and expressions ... 97
11.6.1 Notational conventions ... 97
11.6.2 Expressions ... 97
11.6.3 Arithmetic ... 98
11.6.4 Bit manipulation ... 98
11.6.5 Assignment ... 98
11.6.6 Comparative operators ... 99
11.6.7 Boolean ... 100
11.6.8 Other ... 101
11.6.9 Priority of the operators ... 101
11.7 Statements ... 102
11.7.1 Statement label ... 103
11.7.2 Composite statements ... 103

11.7.3 Expression statement ... 103
11.7.4 Empty statement ... 103
11.7.5 Assert expression ... 103
11.7.6 Break ... 104
11.7.7 Continue ... 104
11.7.8 Do statement while (expression) ... 105
11.7.9 Exit expression ... 105
11.7.10 For (expression 1; expression 2; expression 3) statement ... 105
11.7.11 Goto label ... 106
11.7.12 If (expression) statement 1 else statement 2 ... 106
11.7.13 Return expression ... 107
11.7.14 switch (expression) {case list} ... 107
11.7.15 While (expression) statement ... 108
11.8 Functions ... 109
11.8.1 Function arguments ("call-by-value" versus "call-by-reference") ... 110
11.8.2 Named parameters versus fixed parameters ... 111
11.8.3 Standard values of function arguments ... 112
11.9 Extensions ... 114
11.9.1 Constants ... 114
11.9.2 Device-dependent channel numbers ... 115
11.9.3 Functions ... 116
11.9.3.1 Core functions ... 116
11.9.3.2 Console functions ... 119
11.9.3.3 Mdn functions ... 120
11.9.3.4 Mathematical functions ... 123
11.9.4 Pawn script error codes ... 125
11.10 Example ... 126
11.10.1 Saw-tooth generator ... 126
11.10.2 Weekday designation ... 127
11.10.3 Calculating the flow rate with the table ... 129

11.11 Differences to C ... 131
Chapter 12 Maintenance ... 133
12.1 General maintenance ... 133
12.2 Replacing the fuse ... 133
Chapter 13 Removal/disposal ... 137
Chapter 14 Troubleshooting and repair ... 139
14.1 General problems ... 139
14.2 Log entries and error codes ... 141
14.2.1 Modem error ... 144
14.3 Evaluating the device log ... 147
14.3.1 Evaluating the device log on the myDataweb server ... 147
14.3.2 Evaluating the device log using myDatanetDeviceConfig ... 147
Chapter 15 Spare parts and accessories ... 149
15.1 Internal expansions ... 149
15.2 Antennas ... 149
15.3 Sensors ... 149
15.4 Other accessories ... 150
Chapter 16 Document history ... 151
Chapter 17 Contact information ... 153

6872
6872
6872

Chapter 2 Declaration of conformity

CE

Konformitätserklärung
Declaration of conformity
Déclaration de conformité

Microtronics at

Microtronics Engineering GmbH
Hauptstrasse 7
A-3244 Ruprechtshofen
www.microtronics.at

erklärt in alleiniger Verantwortung, dass das Produkt /
declare under our sole responsibility that our product /
déclare sous sa seule responsabilité que le produit

myDatalogS3x

auf das sich diese Erklärung bezieht, mit den folgenden Normen übereinstimmt /
to which this declaration relates is in comformity with the following standards /
auquel se référe cette déclaration est conforme aux normes

EN61326-1 : 2006-12-01,
EN61010-1 : 2011-08-01

gemäß den Bestimmungen der Richtlinien /
following the provision of Directives /
conformément aux dispositions des Directives

2004/108/EG EMC Directive
2006/95/EG LVD Directive

Ruprechtshofen, 06.07.2012

.....
Hans-Peter Buber, Managing Director

6872
6872
6872

Chapter 3 Technical data

Voltage supply	24 VDC, max.2,9W
Housing	Material: Polycarbonat Weight: 1,5kg Protection class: IP65 Dimensions (WHD): 235 x 240 x 117mm (without antenna)
Operating temperature	-20...+60°C, 15...90%rH non-condensing
Storage temperature	-30...+70°C
Display	LED for indicating the operating mode and error code.
Operation	Solenoid switch for initiating ALOHA transmission mode
USB interface	1 x USB slave for establishing a connection to a PC. The myDatanetDeviceConfig configuration program must be installed on the PC to enable communication with the myDatalogS3 .
Antenna connector	FME
Universal inputs	3 x analogue or digital Modes: • 0 to 20mA: Resolution 6µA, max. 22,5mA, load 100Ω • 4 to 20mA: Resolution 6µA, max. 22,5mA, load 100Ω • 0 to 2V: Resolution 610µV, max. 2,5V, load 10k1 • 0 to 10V: Resolution 8,3mV, max. 32V, load 4,8kΩ • PWM: 1...99%, max. 100Hz, min. impulse length 1ms, load 4,8kΩ • Frequency: 1...1000Hz, 4,8kΩ • Digital: max. 32V, low ‹1,36V, high ›2,73V, load 4,8kΩ • Day counter: min. impulse length 1ms, load 4,8kΩ • Interval counter: min. impulse length 1ms, load 4,8kΩ Additional information is provided in "Technical details about the universal inputs" on page 36.
External temperature sensor	1 x Temperatur sensor TEMP1 (269-TEMP1) (optional equipment)

Outputs	1 x analogue outputs (not galvanically isolated) • Load voltage: 24V • max. load (20mA): 1kΩ • $I_{out\ max}$: 24mA Modes: • 4-20mA • 0-20mA 1 x isolated switch contact (galvanically isolated) • I_{max}: 130mA • U_{max}: 32V • R_{on}: 35Ω • f_{max}: 500Hz Additional information is provided in "Technical details about the outputs" on page 37.
Data memory	Internal flash memory for up to 26360 measurement cycles
Data type	f32 (32 Bit floating point)
Data transmission	By means of GSM/GPRS quad-band modem to the relevant myDataweb server
SIM	The myDatalogS3 is equipped with an integrated SIM chip.
Monthly data volume	t.b.d. at two-minute measurement cycle and 120-minute transmission cycle

Chapter 4 General specifications

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

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

4.1 Translation

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

4.2 Copyright

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

4.3 General descriptive names

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

4.4 Safety instructions

For the connection, commissioning and operation of the myDatalogS3 , the following information and higher legal regulations of the country (e.g. ÖVE), such as valid EX regulations as well as the applicable safety and accident prevention regulations for the respective application case must be observed.

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

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

Important note: *The manufacturer's products that are designed for use outdoors include extensive protection against penetrating moisture and dust. If these products are connected to the power supply or sensors by cables with connectors rather than permanently installed cables, the susceptibility of the connector and socket to moisture and dust penetration is significantly higher. The operator is responsible for protecting the connector and socket against penetrating moisture and dust in a suitable way and complying with local safety regulations.*

4.4.1 Use of the hazard warnings



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



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



CAUTION:
Indicates a potential hazardous situation that can result in minor or moderate injury.

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

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

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

4.4.2 Safety and preventative measures for handling GSM/GPRS modems

Observe the following safety and preventative measures during all phases of installation, operation, maintenance or repair of a GSM/GPRS modem. The manufacturer shall not be held liable if the customer disregards these preventative measures.



CAUTION:
The GSM/GPRS modem connection may not be used in hazardous environments.

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

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

Important note: *No liability shall be assumed at any time and under no condition for the connection via a GSM/GPRS modem for whose use radio signals and networks are utilized, The GSM/GPRS modem must be switched on and be operated in an area where sufficient signal strength is present.*

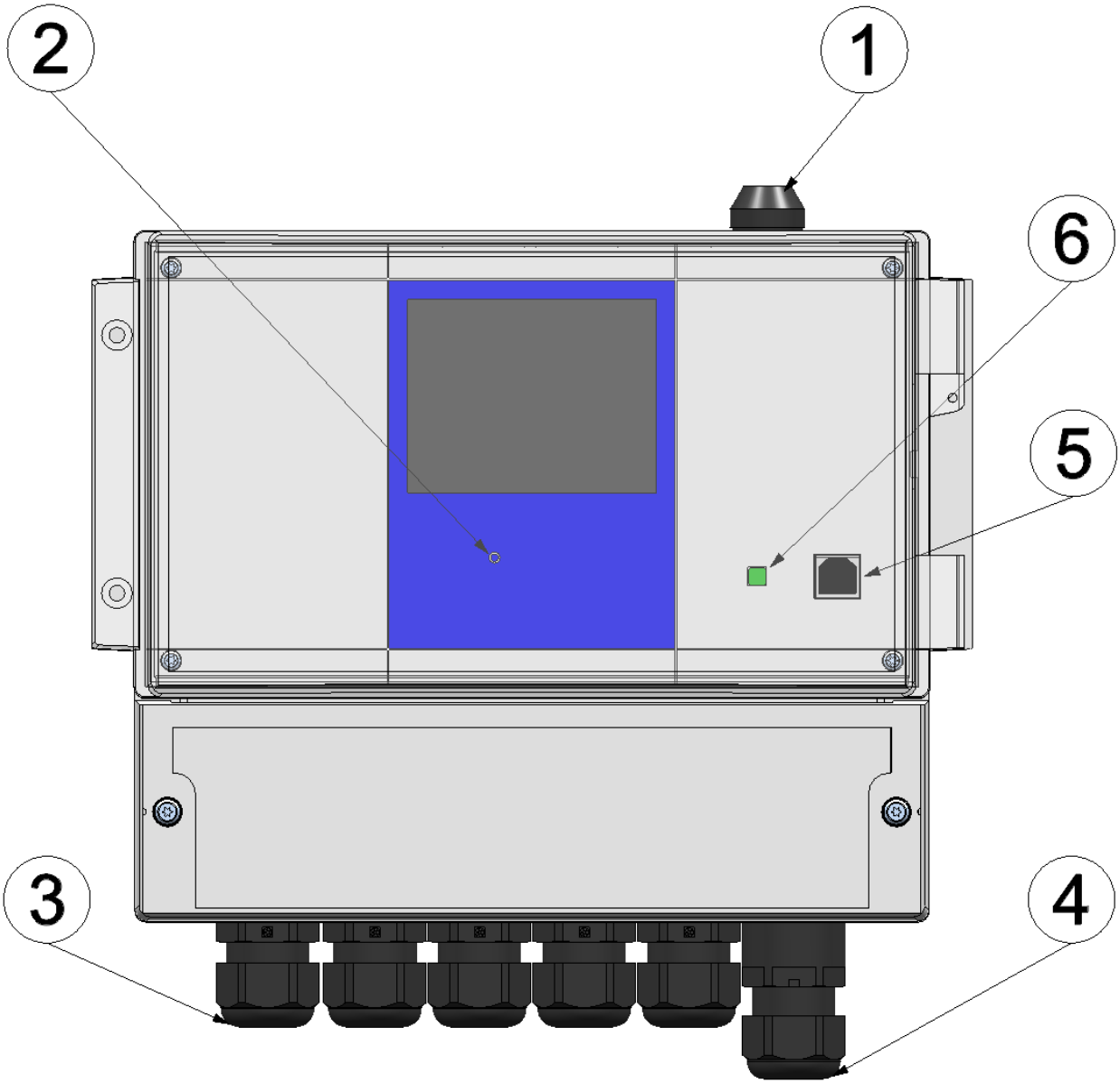
4.4.2.1 Safety and precautionary measures for the GSM/GPRS modem installation

- This device may only be installed by a trained technician who applies the recognised installation practices for a radio frequency transmitter including the of external antennas.
- The device may not be operated in hospitals and/or near medical equipment such as heart pacemakers or hearing aids.
- The device may not be operated in highly flammable areas such as petrol filling stations, fuel storage sites, chemical factories and explosion sites.
- Do not operate the device in the vicinity of flammable gases, vapours or dusts.
- Do not subject the device to strong vibrations or impacts.
- The GSM/GPRS modem can cause interference if it is located near television equipment, radios or computers.
- Do not open the GSM/GPRS modem. Any modification to the equipment is prohibited and shall lead to loss of the operating licence.
- The use of GSM services (SMS messages/data communication/GPRS etc.) may lead to additional costs. The user alone is responsible for any damages and costs resulting through this.
- Do not install the device in any other manner than the one described in the operating instructions. An improper use will invalidate the warranty.

4.4.2.2 Safety measures for installing the antenna

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

4.5 Overview



Overview myDatalogS3

1	Antenna connection	4	M20x1.5 cable screw connection with pressure compensation (cable diameter of 6-13mm)
2	Solenoid switch	5	USB interface
3	M20x1.5 cable screw connection (cable diameter of 5-10mm)	6	Status LED

4.6 Intended use

The stationary measurement device is used to record analogue and digital signals. The device requires a continuous power supply. The measured and recorded data is stored on a non-volatile memory medium. This stored data is sent via the mobile network to a central server for further processing. The device is equipped with an integrated SIM card for this purpose. The maximum permissible limit values specified in chapter "Technical data" on page 13 must be observed. The manufacturer shall not be liable for any operational cases that deviate from these limit values and have not been approved by the manufacturer in writing.

> ***Note:*** *This device is exclusively intended to be used for the purposes as described above. Any other use or use beyond what is specified or a modification of the device shall be deemed to be not for the intended purpose and is not permitted without the express written consent of the manufacturer. The manufacturer shall not be held liable for any damages that may result from such unauthorised use or modification. The operator alone bears the associated risk.*
>
> ***Note:*** *The manufacturer is not liable for data loss of any kind if the device is damaged and data is not able to be stored correctly.*
>
> ***Note:*** *The integrated SIM chip provides a mobile communications connection to a variety of international service providers. In order to be able to utilise all functions of the device, you must ensure that the device is located in the service area of one of these service providers. You can find a list of all supported countries and associated service providers under www.microtronics.at/footprint. A Managed Service contract with Microtronics Engineering GmbH is required for use of the mobile data transmission (see www.microtronics.at/managedservice). This includes the provisioning of the mobile communications connection via the network of the service provider included in the above-mentioned list.*

4.7 General product information

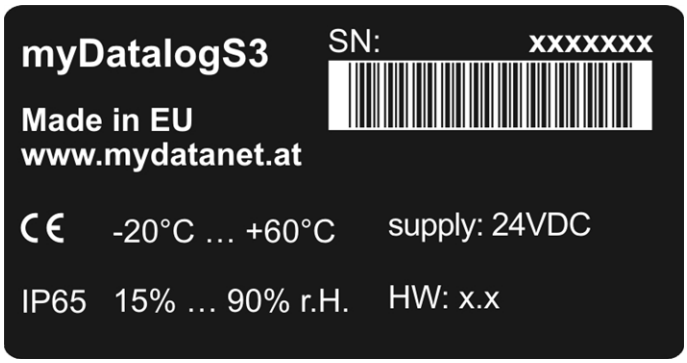
The device is a stationary device for recording and transmitting analogue and digital signals. There are 3 universal inputs available, which can be operated in various analogue and digital modes. A one wire bus is also available for connecting an external temperature sensor. 2 output channels are also available. One of these is a 0/4-20 mA analogue output for which the output values are provided wirelessly from a central location. The other is a galvanised isolated switch contact that can be configured so that it can be switched by the device itself prior to a measurement (to supply a sensor) or so that it can be switched wirelessly from a central location. All input measurement data is temporarily saved to an internal data memory along with the output statuses and wirelessly transmitted to a central location at freely selected intervals. This connection can also be used to configure the device. The device is equipped with an integrated SIM card.

4.8 Device labelling

The specifications in this user manual apply exclusively to the myDatalogS3 device type. The type plate is located on the right side of the device and contains the following specifications:

- Type designation
- CE marking
- Protection class
- Environmental conditions during operation
- Serial number
- Voltage supply specifications
- 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 myDatalogS3

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

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

4.9 Installation of replacement and wear parts

Please be advised that replacement and accessory parts that have not been supplied by the manufacturer have also not been inspected or approved by it. Installing or using such products and similar can constructively modify the specified properties of the device in a negative way. The manufacturer shall not be liable for any damages that arise from the use of non-original parts and non-original accessory parts.

4.10 Storage of the product

For safekeeping of the myDatalogS3 , ensure that all of the relevant data was transferred to the myDataweb server. If necessary initiate the ALOHA transmission mode using the solenoid contact (see "ALOHA transmission mode" on page 23) and then check again that all of the relevant data has

been transmitted. Then disconnect the device from the power supply. If possible, switch off the power supply before disconnecting the cables of the X1 clamp in the clamping space (see"Connecting the sensors, actuators and power supply" on page 31). The remaining cables and the antenna can then be removed. Store the myDatalogS3 in its original packaging.

The configuration and most recently determined data are retained. The myDatalogS3 includes a buffer battery that ensures that the system time is maintained. However, if the device is stored for a very long period without a power supply, the buffer battery may discharge fully causing the system time to fail. Even if the buffer battery has not fully discharged during storage, the system time may no longer be synchronised with the time base of the myDataweb server. It is therefore recommended to synchronise the time when the device is placed back into operation, so that the measurement and log data time stamps are correct. This occurs automatically either when first connected to the myDataweb server or when the myDatalogS3 is connected to the PC. The myDatanetDeviceConfig configuration program is required for this purpose.

4.11 Obligation of the operator



WARNING:
In the EEA (European Economic Area), the national implementation of the framework directive (89/391/EEC) as well as the associated specific directives and from these in particular, the directive (89/655/EEC) about the minimum safety and health requirements for use of work equipment by workers at work, each in their respective version are to be complied with.

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

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

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

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

6872
6872
6872

Chapter 5 Functional Principle

5.1 ALOHA transmission mode

ALOHA transmission mode is a special connection mode whereby the myDatalogS3 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 70).

There are two options for initiating the ALOHA transmission mode:

- Directly on the device using the solenoid switch (see "Solenoid switch" on page 43)
- Via the myDataweb server if the "interval and wakeup" connection type has been selected (see "Basic settings" on page 70).



A speech bubble in the measurement device list (see "myDatanet Server 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 "myDatanet Server Manual " 206.886), It includes the internal "voltage" and "GSM level" measurement values, as well as the measurement values of the 3 universal inputs. The measurement values for the last thirty minutes are always shown in the ALOHA data window, and in some cases also include those from a previous ALOHA transmission. The ALOHA data is generated every three seconds independently of the normal measurement value generation 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 to clarify the recording, measurement and burst interval (in conjunction with the decay)" on page 71), 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 "myDatanet Server Manual " 206.886).

5.2 Selection of the GSM network

The GSM network to which the device should register must be selected, as the myDatalogS3 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 myDatalogS3 logs itself into a GSM network:

- PowerOn
- Expiration of the transmission cycle if the "Interval" connection type was selected (see "Basic settings" on page 70).
- Activation of the "Interval & Wakeup" or "Online" connection type (see "Basic settings" on page 70)

1. The myDatalogS3 checks whether any errors occurred during the last login attempt.

   No errors:

   The myDatalogS3 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 24.*

5.2.1 Ban list

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

If a "BEARER GPRS FAILED" error occurs (see "Modem error" on page 144), 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 24).

5.3 Script eradication

5.3.1 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 80). 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.*

5.3.2 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 125).

6872
6872
6872

Chapter 6 Storage, delivery and transport

6.1 Inspection of incoming deliveries

Please check the shipment immediately after receiving for completeness and intactness. Immediately report any discovered transport damages to the delivering carrier. Also notify the manufacturer in writing about this without delay. Report any incompleteness in the delivery to the responsible representative or directly to the company headquarters of the manufacturer within 2 weeks (see "Contact information" on page 153).

Note: *Any claims received thereafter will not be acknowledged!*

6.2 Scope of supply

The standard scope of delivery of themyDatalogS3 (206.254) includes:

- myDatalogS3
- Assembly clips for the top-hat rail assembly
- Antenna 900 FME (206.801)
- Managed Service SIM (installed in the device)
- USB A/B Kabel (206.601)
- Replacement fuse MDL8 (206.312)
- MDN Magnet (206.803)
- myDatanet Tool Pen (206.646)

Additional accessories such as assembly sets, antennas, the power supply, sensors, etc. are dependent on the order specifications. Please check these against the delivery slip.

6.3 Storage

The following storage conditions must be observed:

myDatalogS3	Storage temperature	-30...+70°C
	Humidity	15...90%rH

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

6.4 Transportation

The myDatalogS3 is designed for rough industrial applications. Nevertheless, it should not be subjected to heavy shocks, bumps, impacts or vibrations. The original packaging must always be used for transport.

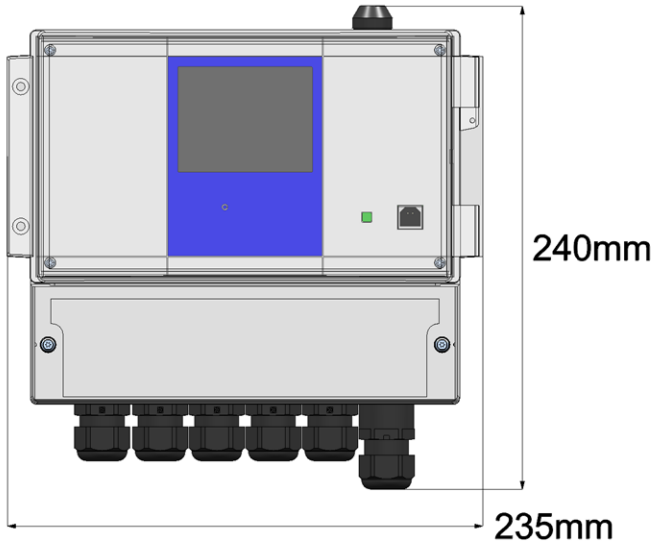
6.5 Return

Every return must be accompanied by a fully field-out return form. This return form is available in the service area of the myDataweb-Server. An RMA number is mandatory for any returns and can be obtained from the Support & Service Centre (see "Contact information" on page 153). The return shipment of the myDatalogS3 must occur in the original packaging and with freight and insurance paid to Microtronics Engineering GmbH (see "Contact information" on page 153). Insufficiently cleared return shipments will otherwise not be accepted!

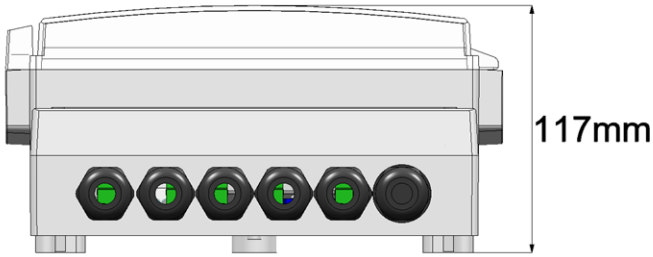
Chapter 7 Installation

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

7.1 Dimensions



Dimensions: Width and height



Dimensions: Depth

7.2 Installing the myDatalogS3

Important note:

- *Ensure the installation is completed properly.*
- *Comply with the existing legal and/or operational directives.*
- *Improper handling can cause injuries and/or damage to the devices.*

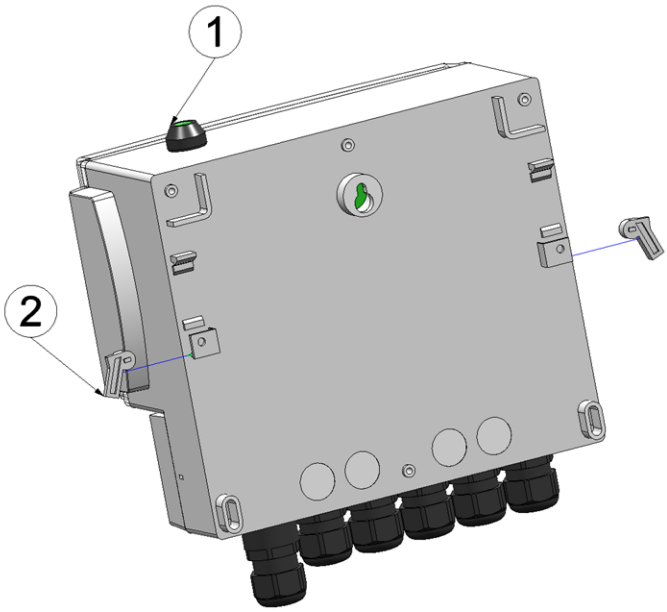
The installation site must be selected according to specific criteria. The following conditions must be avoided in any case:

- Direct sunlight
- Objects that radiate intense heat (maximum ambient temperature: -20...+60°C)
- Objects with a strong electromagnetic field (frequency converter or similar)
- Corrosive chemicals or gases
- Mechanical impacts
- Direct installation on foot or road ways
- Vibrations
- Radioactive emissions

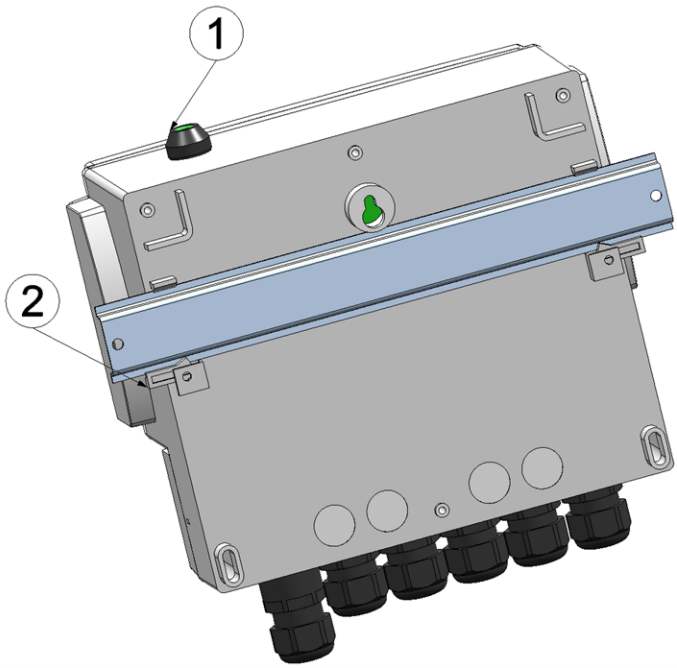
Note: *Leave sufficient space at the upper end to mount the antenna. The space required depends on the antenna used. Approx. 15 cm of space must be left beneath the device for the cable connections. Further information about the installation dimensions can be found in the relevant sub-chapter.*

7.2.1 Top-hat rail assembly

The assembly clips required for the top-hat rail assembly, are included in the scope of supply of the myDatalogS3 .



Top-hat rail assembly step 1



Top-hat rail assembly step 2

1 myDatalogS3	**2** Assembly clips for the top-hat rail assembly (included in the scope of supply of the myDatalogS3)

1. First of all install the assembly clips (2) on the myDatalogS3 (see "Top-hat rail assembly step 1 " on page 30). Please note that the assembly clips are not identical. One is designed for the right and the other for the left side of the housing.
2. Place the housing hooks on the top edge of the top-hat rail (see "Top-hat rail assembly step 2" on page 30). Then fix the myDatalogS3 by folding the assembly clips (2) upwards.

7.3 Safety instructions for the cabling

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

When connections are made to the myDatalogS3 , the following warnings and information must be observed, in addition to the warnings and information found in the individual chapters on the installation. Further safety information is included in "Safety instructions" on page 15.

7.3.1 Information on preventing electrostatic discharges (ESD)

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

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

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

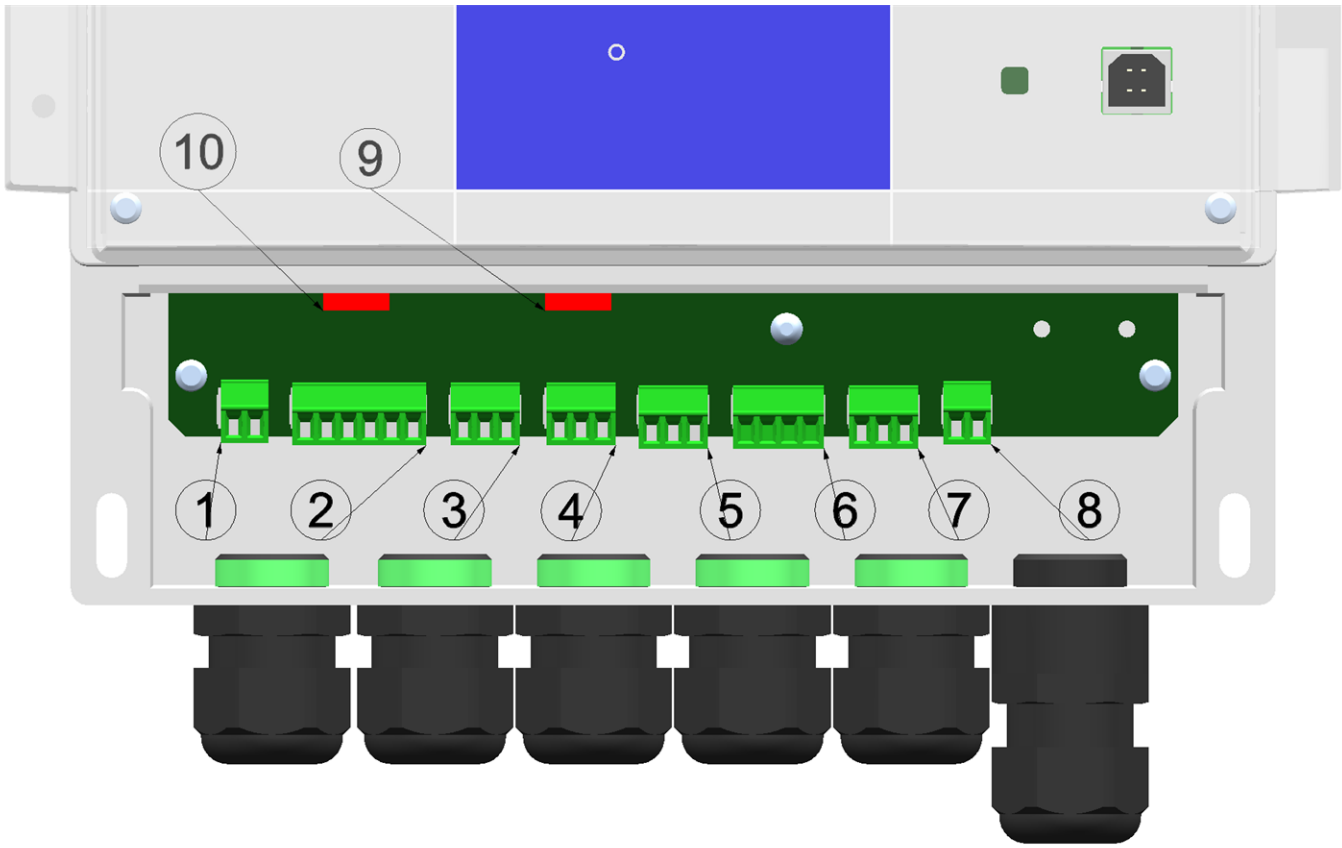
7.4 Electrical installation

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

7.4.1 Connecting the sensors, actuators and power supply

Important note:

- *All cabling work must be performed in the de-energised state.*
- *Ensure installation is completed correctly.*
- *Comply with the existing legal and/or operational directives.*
- *Improper handling can cause injuries and/or damage to the devices.*
- *Run all data and power cables so that they do not pose a trip hazard and ensure that cables do not have any sharp bends.*
- *The myDatalogS3 must not be operated in the field with the lids open.*
- *To ensure the housing is properly sealed, each of the cable screw connections must only hold a single cable.*
- *All unused cable screw connections on the myDatalogS3 must be sealed watertightly using the blind plugs supplied.*
- *Observe the specifications regarding the maximum cable lengths.*



Connecting the sensors, actuators and power supply

1	Device power supply (X1)	6	Output 1: isolated switch contact (X6), max. cable length: 2,7m
2	RS232 interface[1)] (X2), max. cable length: 2,7m	7	Connection for the ext. temperature sensor (X7), max. cable length: 1m
3	Universal input 1 (X3), max. cable length: 2,7m	8	Output 2: analogue output (X8), max. cable length: 2,7m
4	Universal input 2 (X4), max. cable length: 2,7m	9	Dip switch (dip switch 1) for activating/deactivating the additional 680Ω resistances to increase the load of the universal inputs.
5	Universal input 3 (X5), max. cable length: 2,7m	10	Dip switch (dip switch 2), reserved for extensions

[1)]The RS232 interface is only used to provide the values written in the display register (see "Using the display registers" on page 87) via the instruction list (see "Instruction list" on page 75).

Assignment of the clamp to the device power supply (X1)

1	24V	Supply voltage: 24 VDC, max. 2,9W
2	GND	Mass

Assignment of the clamp to the RS232 interface (X2)

1	24V	24V sensor supply
2	TX	RS232 TX line
3	RX	RS232 RX line
4	B	Reserved for extensions
5	A	Reserved for extensions
6	GND	Mass

Assignment of the clamp to universal input 1 (X3)

1	24V	24V sensor supply for universal input 1
2	IN1	Universal input 1
3	GND	Mass

Assignment of the clamp to universal input 2 (X4)

1	24V	24V sensor supply for universal input 2
2	IN2	Universal input 2
3	GND	Mass

Assignment of the clamp to universal input 3 (X5)

1	24V	24V sensor supply for universal input 3
2	IN3	Universal input 3
3	GND	Mass

Assignment of the clamp to the isolated switch contact (X6)

1	24V	24V sensor supply
2	NO	Closing contact (normally opened)
3	CC	Root (common)
4	GND	Mass

Assignment of the clamp to the connection of the ext. temperature sensor (X7)

1	5V	5V sensor supply
2	1W	Data line
3	GND	Mass

Assignment of the clamp to analogue output (X8)

1	mA out	0/4-20mA output signal
2	GND	Mass

Dip switch (dip switch 1) for activating/deactivating the additional 680Ω resistances to increase the load of the universal inputs

S1	680Ω additional load for universal input 1
S2	680Ω additional load for universal input 2
S3	680Ω additional load for universal input 3
S4	Reserved for extensions

Dip switch (dip switch 2), reserved for extensions

S1	Reserved for extensions
S2	Reserved for extensions
S3	Reserved for extensions
S4	Reserved for extensions

1. Remove both of the screws that secure the housing compartment cover. Now open the housing compartment of the myDatalogS3 .

   Important note: *In the event of adverse weather conditions including rain or in a location where water can penetrate from above, suitable measures must be implemented to protect the device from penetrating moisture when the housing compartment cover is open.*

2. Then connect your sensors and actuators with the inputs and outputs. Ensure that no current is present! The cables for the power supply of the myDatalogS3 should be de-energised when being connected to the supply connector.

   Important note: *Only a single cable must be threaded through the cable screw connections to ensure the seal of the housing is not jeopardised.*

3. Tighten the cable screw connections to secure the cables.

4. Affix blind plugs to all of the cable screw connections that are not required.

   Important note: All unused cable screw connections on the myDatalogS3 must be sealed watertightly using the blind plugs supplied. Otherwise the degree of protection for the entire device is not guaranteed and the manufacturer's warranty is void.

5. Connect the antenna (see "Connecting the GSM antenna" on page 35).

6. Close the housing compartment cover.

   Important note: Ensure that the seals are clean and intact before closing the housing compartment cover. Remove any impurities and/or dirt. The manufacturer shall not be liable for any damage to the device caused by leaky or faulty seals.

7. Switch on the 24 VDC power supply of the myDatalogS3 . The status LED should then start to flicker (see "Status LED" on page 44), indicating that a connection is being established.

The following step is not mandatory.

8. Check whether the connection to the myDataweb has worked correctly (see "Testing communication with the device" on page 40).

7.4.2 Connecting the GSM antenna

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

The standard antenna is directly attached to the antenna connector (see "Overview" on page 18) of the myDatalogS3 . In the event of a low radio signal strength, you can use the Flat antenna Smart Disc FME-F (206.814).

If the distance between the antenna position and the myDatalogS3 is too great, you can use a 5 m Extension cable for Antenna 900 FME (206.805).

1. Ensure that the myDatalogS3 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 connection of the myDatalogS3 (see "Overview" on page 18)

   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 myDatalogS3 back on.

The following step is not mandatory.

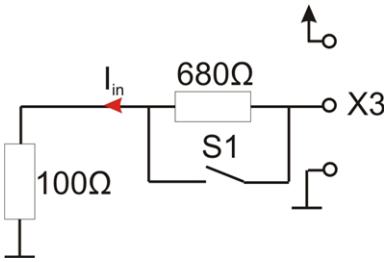
5. Check whether the connection to the myDataweb has functioned correctly (see "Testing communication with the device" on page 40).

7.4.3 Technical details about the universal inputs

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

A 680Ω resistance can be connected as an additional load for the relevant universal input via dip switch 1 (see "Connecting the sensors, actuators and power supply" on page 31). This is provided to protect the universal input in case the voltage at the input in 0/4 to 20mA mode is too high because of a sensor error. If you contemplate activating the additional load, ensure that your sensor is able to drive this increased load.

> ***Important note:*** *The additional load must only be activated for universal inputs that are being operated in 0/4 to 20mA mode. The additional load will cause a measurement error with all of the other operating modes.*



Schematic diagram of universal input 1 in 0/4 to 20mA mode

7.4.3.1 0/4 to 20mA mode

Note: *Above 22,5mA, the relevant input becomes highly resistive (safety shutdown to prevent damage to the universal input).*

Resolution	6µA
I_{max}	22,5mA
Load	100Ω

7.4.3.2 0 to 2V mode

Resolution	610µV
U_{max}	2,5V
Load	10k1

7.4.3.3 0 to 10V mode

Resolution	8,3mV
U_{max}	32V
Load	4,8kΩ

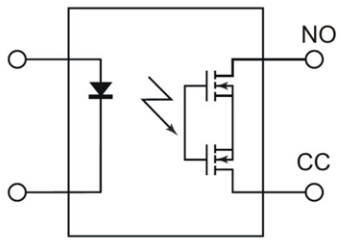
7.4.3.4 Standard digital modes (PWM, frequency, digital, day counter, impulse counter)

General	U_{max}	32V
	Low	‹1,36V
	High	›2,73V
	Load	4,8kΩ
PWM	Measurement range	1...99%
	f_{max}	100Hz
	Minimum impulse length	1ms
Frequency	Measurement range	1...1000Hz
Day and impulse counter	Minimum impulse length	1ms

7.4.4 Technical details about the outputs

7.4.4.1 Output 1: Isolated switch contact (NO, CC)

Important note: The user must ensure that the current through the isolated switch contact does not exceed 130mA.

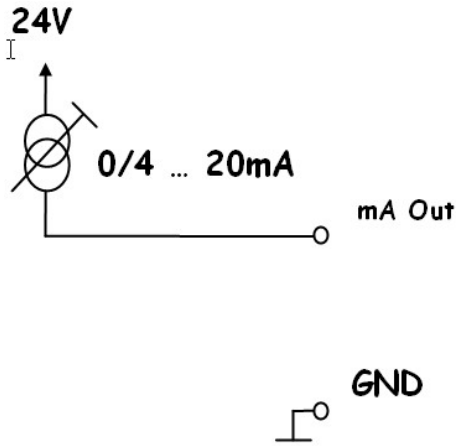


Replacement circuit diagram for the isolated switch contact

I_{max}	130mA
U_{max}	32V
R_{on}	35Ω
f_{max}	500Hz

7.4.4.2 Output 2: Analogue output (mA Out, GND)

Note: *The analogue output of the myDatalogS3 is an active, not galvanically isolated current output.*



Replacement circuit diagram for the analogue output

Load voltage	24V
max. load (20mA)	1kΩ
$I_{out\ max}$	24mA

Chapter 8 Initial Start-Up

8.1 User information

Before you connect the myDatalogS3 and place it into operation, you must observe and comply with the following user information!

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

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

Read this manual carefully and completely in order to ensure the proper functioning of the myDatalogS3 .

Please contact our technical department if anything is unclear or if you encounter difficulties with regard to installation, connection or configuration (see "Contact information" on page 153).

If necessary, please also refer to the manuals of the accessory parts when commissioning the complete system. These manuals are included in the scope of delivery of the accessory parts.

8.2 General principles

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

Utilise the manual to familiarise yourself with the operation of the myDatalogS3 and the input screens of the myDataweb-Server before you begin with the configuration.

8.3 Placing the system into operation

> ***Note:*** *It is recommended that the myDatalogS3 is first placed into operation in the office before mounting the device permanently at the place of use. During this process, you should set a site for the later operation on the myDataweb server (see "myDatanet Server Manual " 206.886) and determine a site configuration (see "Site configuration" on page 45). Take the opportunity to get to know the functions of the device in a stable environment. You can also use suitable test signals to simulate the sensors to establish the optimum configuration of the myDatalogS3 prior to its actual first use. This reduces the amount of time required for on-site installation to a minimum.*

1. Once you have completed all the steps described in the chapter "Connecting the sensors, actuators and power supply" on page 31, the myDatalogS3 is ready for operation and should have made its first connection to the myDataweb server.
2. Create a site for the operation on the myDataweb server (see "myDatanet Server Manual " 206.886).

3. Configure the created site according to your requirements (see "Site configuration" on page 45).
4. Link the myDatalogS3 with the created site (see "Site" on page 45).
5. Activate the ALOHA transmission mode (see "ALOHA transmission mode" on page 23) so that the site configuration is transmitted to the myDatalogS3 .

8.4 Testing communication with the device

1. Create a site for the operation on the myDataweb server (see "myDatanet Server Manual " 206.886).
2. Configure the created site according to your requirements (see "Site configuration" on page 45).
3. Link the myDatalogS3 with the created site (see "Site" on page 45).
4. Initiate the ALOHA transmission mode (see "ALOHA transmission mode" on page 23) so that the configuration of the site is transmitted to the myDatalogS3 .
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.



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

6. Stop the ALOHA transmission mode by clicking the cross in the speech bubble with the "Aloha" inscription or wait for the duration of the ALOHA transmission mode. This period can be set in the basic settings (see "Basic settings" on page 70) of the site configuration. The default setting is 10 minutes.
7. Then wire up the sensors (see "Connecting the sensors, actuators and power supply" on page 31) and restart the ALOHA transmission mode.

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

8. Check the incoming data in the ALOHA data window of the myDataweb server, which can be accessed by clicking on the speech bubble with the "Aloha" inscription (see "myDatanet Server Manual " 206.886). Particular attention must be paid to the internal "GSM level" and "voltage" measurement values.



Note: *Additional explanation about evaluating the "GSM level":*

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

Note: *Additional explanation about evaluating the "voltage":*

The displayed voltage should only deviate slightly from the supply voltage applied to the device.

6872
6872
6872

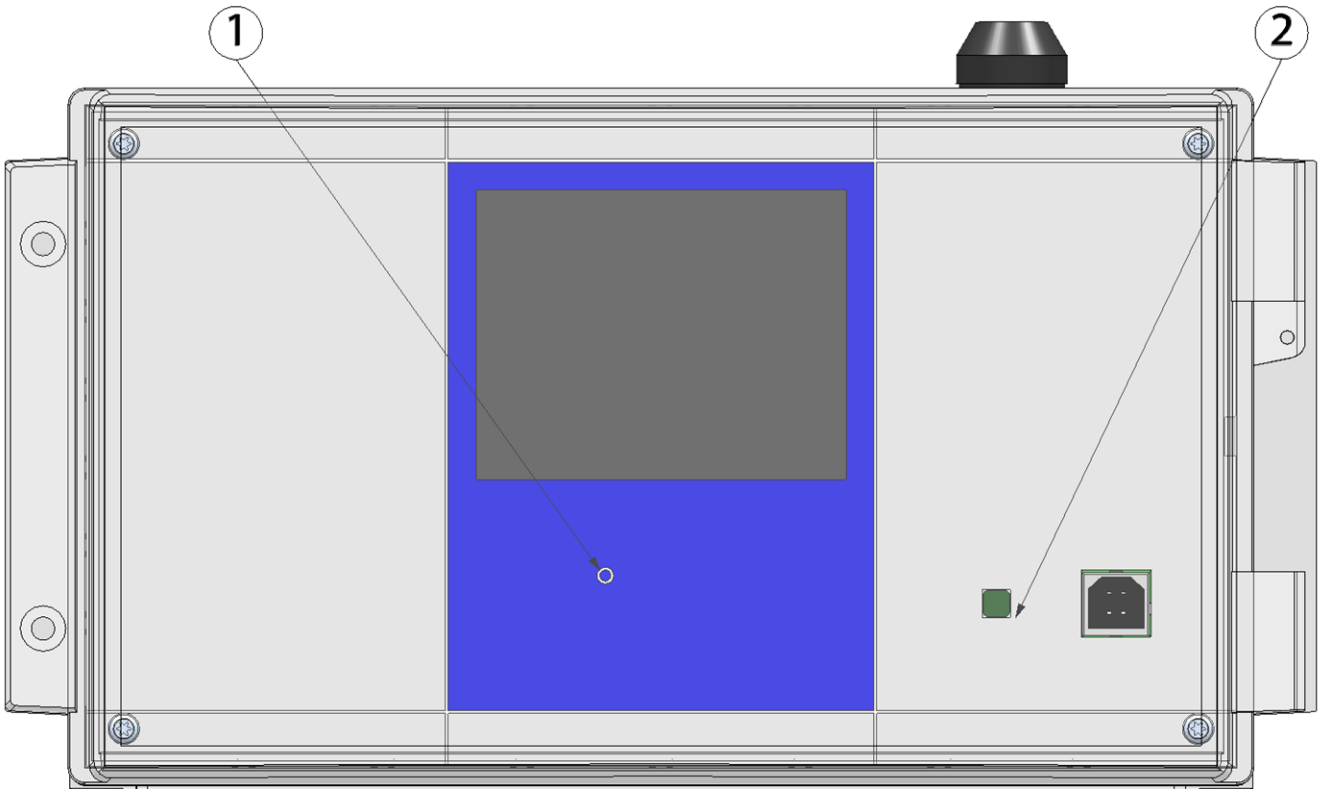
Chapter 9 User interfaces

The configuration of the myDatalogS3 is carried out via the web interface on the myDataweb server (see "User interface on the myDataweb-Server" on page 45), which your responsible sales partner will provide to you.

9.1 User interface on the myDatalogS3

9.1.1 Operating elements

Important note: In the event of adverse weather conditions including rain or in a location where water can penetrate from above, suitable measures must be implemented to protect the device from penetrating moisture when the housing cover is open.



Operating elements that are accessible when the housing cover is open

1 Solenoid switch	2 Status LED

9.1.1.1 Solenoid switch

The MDN Magnet (206.803) included in the scope of delivery is required for operating the solenoid switch. The solenoid switch can be used to initiate the ALOHA transmission mode or to instruct the myDatalogS3 to immediately transmit the error/status code.

Operation by the user	Device response	Operation after releasing the solenoid switch
Press briefly (approx. one second)	Status LED illuminates	Error/status code is displayed (see "Status LED" on page 44)
Press and hold for five seconds	Status LED flashes three times and then remains on	ALOHA transmission mode

9.1.1.2 Status LED

The status LED is used both to display the error/status codes and to indicate the current operating state. If the ALOHA transmission mode was activated or the power supply was provided (PowerOn), the status LED shows the current operating state for ten minutes. During these ten minutes, the error/status codes are transmitted every three seconds.

Error/status codes

Flashing code	Description	Solution/cause
0x	Transport lock (GPRS OFF, measurement OFF)	If the ALOHA transmission mode is initiated by the solenoid switch , the myDatalogS3 switches back into "RUN" mode (GPRS ON, measurement ON).
1x	Last connection OK	---
2x	Last transmission faulty	Try again later
4x	Standby (GPRS ON, measurement OFF)	see "transport lock"
6x	Offline (GPRS OFF, measurement ON)	see "transport lock"
7x	Network block/no matching provider	• Improve position of the antenna • Check whether the device is located in the coverage area of a service provider that is supported by the integrated SIM chip (www.microtronics.at/footprint) • Lift the network block (see "myDatanetDeviceConfig Manual " 206.887)
8x	No GSM network	• Try again later • Improve position of the antenna
10x	No GPRS connection	Improve position of the antenna
11x	No myDataweb server accessible	• Check whether port 51241 is enabled on the myDataweb server • Try again later
12x	Faulty SIM chip	Contact support

Operating states

Status LED	Description
Flickering	Establishing connection
Lights up	GPRS connection established or solenoid switch actuated
Off	Normal measuring operation according to configuration until the next transmission

9.2 User interface on the myDataweb-Server

9.2.1 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 "myDatanet Server Manual " 206.886).

9.2.1.1 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!)

9.2.1.2 Comments

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

9.2.1.3 Control

Script type	Off	Script deactivated
	IL	Activates script processing and informs the myDatalogS3 that the script is an "Instruction list".
	Pawn	Activates script processing and informs the myDatalogS3 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 myDatalogS3 on the myDataweb server (to create a template, see "myDatanet Server 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 myDatalogS3 .	

9.2.1.4 Measurement channels

9.2.1.4.1 Basis

Title 1-3

Freely selectable channel title for the universal inputs

Title 1Wire

Freely selectable channel title for the external temperature sensor

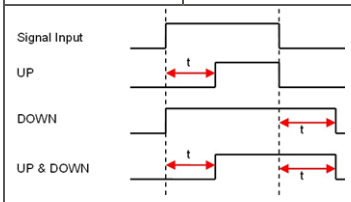
Mode

Basic settings for the measurement channel

Universal inputs (digital modes)	*off*	*---*	*Measurement channel deactivated*
	Digital	*Invert*	*Inverts the input signal*
	Cnt.Day	*Impulse*	*Count value of an impulse in the measuring unit*
		Max	*Defines the upper scale end of the pointer instruments*
		Unit	*String that is used as a 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*
	Cnt.Intervl.	*Impulse*	*Count value of an impulse in the measuring unit*
		Max	*Defines the upper scale end of the pointer instruments*
		Unit	*String that is used as a 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*
	Freq	*Factor*	*Factor by which the input signal is multiplied*
		Min	*Defines the lower scale end of the pointer instruments*
		Max	*Defines the upper scale end of the pointer instruments*
		Unit	*String that is used as a 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*
	PWM	*0%*	*Start of the measurement range in the measuring unit*
		100%	*End of the measurement range in the measuring unit*
		Unit	*String that is used as a 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*

Universal inputs *(analogue modes)*	*0-20mA*	*0%*	*Start of the measurement range in the measuring unit*
		100%	*End of the measurement range in the measuring unit*
		Unit	*String that is used as a 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*
	4-20mA	*0%*	*Start of the measurement range in the measuring unit*
		100%	*End of the measurement range in the measuring unit*
		Unit	*String that is used as a 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*
	0-2V	*0%*	*Start of the measurement range in the measuring unit*
		100%	*End of the measurement range in the measuring unit*
		Unit	*String that is used as a 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*
	0-10V	*0%*	*Start of the measurement range in the measuring unit*
		100%	*End of the measurement range in the measuring unit*
		Unit	*String that is used as a 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*
Ext. temperature sensor	*off*	*---*	*Measurement channel deactivated*
	on	*Min*	*Defines the lower scale end of the pointer instruments*
		Max	*Defines the upper scale end of the pointer instruments*
		Unit	*Selection of the temperature unit used by all of the server display elements*
		Decimal places	*Number of decimal places that are used by all of the server display elements*

9.2.1.4.2 Config

<table>
<tr><td rowspan="8">Universal inputs
(digital modes)</td><td>off</td><td>---</td><td colspan="2">---</td></tr>
<tr><td rowspan="7">Digital</td><td>Filter time</td><td colspan="2">Minimum signal length for the signal detection</td></tr>
<tr><td rowspan="5">Decay</td><td colspan="2">Temporal function in the measurement cycle</td></tr>
<tr><td>Off</td><td>Decay deactivated</td></tr>
<tr><td>up</td><td>Minimum signal length for x seconds with a rising edge</td></tr>
<tr><td>down</td><td>Minimum signal length for x seconds with a falling edge</td></tr>
<tr><td>up&down</td><td>Minimum signal length for x seconds on both edges</td></tr>
<tr><td colspan="2">
</td></tr>
<tr><td>Time</td><td colspan="2">Time x, for which the decay modes "up", "down" and "up&down" are used</td></tr>
</table>

Universal inputs (Counter modes)	Cnt.Day	Filter time	Minimum signal length for the signal detection	
		Reset at	Reset time of the day counter	
	Cnt.Intervl.	Filter time	Minimum signal length for the signal detection	
		Decay	Temporal function in the measurement cycle	
			off	Decay deactivated
			min	The minimum of the last x measurement values is recorded.
			max	The maximum of the last x measurement values is recorded.
			avg	The arithmetic mean of the last x measurement values is recorded.
			med	The median of the last x measurement values is recorded.
			rms	The root mean square of the last x measurement values is recorded.
		Time	Time window for the decay. Up to a maximum of 60 measurement valuescan be used for the calculation (e.g. 1 channel: 60 measurement values; ... 4 channels: 15 measurement values each). To calculate the number of considered measurement values see "Example to clarify the recording, measurement and burst interval (in conjunction with the decay)" on page 71.	

Universal inputs (Frequency mode 1/2)	Freq (1/2)	Filter time	Minimum signal length for the signal detection	
		Decay	Temporal function in the measurement cycle	
			off	Decay deactivated
			min	The minimum of the last x measurement values is recorded.
			max	The maximum of the last x measurement values is recorded.
			avg	The arithmetic mean of the last x measurement values is recorded.
			med	The median of the last x measurement values is recorded.
			rms	The root mean square of the last x measurement values is recorded.
		Time	Time window for the decay. Up to a maximum of 60 measurement valuescan be used for the calculation (e.g. 1 channel: 60 measurement values; ... 4 channels: 15 measurement values each). To calculate the number of considered measurement values see "Example to clarify the recording, measurement and burst interval (in conjunction with the decay)" on page 71.	
		Hold	Hold the last valid measurement value for x measurement cycles	
			off	Function deactivated
			1-5	Number of measurement cycles for which the measurement value is held until the error value is issued
			on	In the event of an error, the last valid measurement value is held until a new valid measurement value is present.

Universal inputs (Frequency mode 2/2)	Freq (2/2)	Overflow	Procedure in the event of measurement range violations	
			Ignore	The measurement value is calculated beyond the range limits.
			Silent cutoff	The measurement value is truncated at the range limits.
			Overflow	• If the measurement value is below the lower limit, the error value "UF" (underflow) is issued. • If the measurement value is above the upper limit, the error value "OF" (overflow) is issued.
			NAMUR borders	• If the measurement value is below the lower limit, the error value "UF" (underflow) is issued. • If the measurement value is above the upper limit, the error value "OF" (overflow) is issued.

Universal inputs (PWM mode 1/2)	PWM (1/2)	Filter time	Minimum signal length for the signal detection	
		Decay	Temporal function in the measurement cycle	
			off	Decay deactivated
			min	The minimum of the last x measurement values is recorded.
			max	The maximum of the last x measurement values is recorded.
			avg	The arithmetic mean of the last x measurement values is recorded.
			med	The median of the last x measurement values is recorded.
			rms	The root mean square of the last x measurement values is recorded.
		Time	Time window for the decay. Up to a maximum of 60 measurement valuescan be used for the calculation (e.g. 1 channel: 60 measurement values; ... 4 channels: 15 measurement values each). To calculate the number of considered measurement values see "Example to clarify the recording, measurement and burst interval (in conjunction with the decay)" on page 71.	
		Hold	Hold the last valid measurement value for x measurement cycles	
			off	Function deactivated
			1-5	Number of measurement cycles for which the measurement value is held until the error value is issued
			on	In the event of an error, the last valid measurement value is held until a new valid measurement value is present.

Universal inputs (PWM mode 2/2)	PWM (2/2)	Overflow	Procedure in the event of measurement range violations	
			Ignore	The measurement value is calculated beyond the range limits.
			Silent cutoff	The measurement value is truncated at the range limits.
			Overflow	• If the measurement value is below the lower limit, the error value "UF" (underflow) is issued. • If the measurement value is above the upper limit, the error value "OF" (overflow) is issued.
			NAMUR borders	• If the measurement value is below the lower limit, the error value "UF" (underflow) is issued. • If the measurement value is above the upper limit, the error value "OF" (overflow) is issued.

Universal inputs (0-20mA mode 1/2)	0-20mA (1/2)	Filter time	Temporal averaging for signal smoothing	
		Decay	Temporal function in the measurement cycle	
			off	Decay deactivated
			min	The minimum of the last x measurement values is recorded.
			max	The maximum of the last x measurement values is recorded.
			avg	The arithmetic mean of the last x measurement values is recorded.
			med	The median of the last x measurement values is recorded.
			rms	The root mean square of the last x measurement values is recorded.
		Time	Time window for the decay. Up to a maximum of 60 measurement valuescan be used for the calculation (e.g. 1 channel: 60 measurement values; ... 4 channels: 15 measurement values each). To calculate the number of considered measurement values see "Example to clarify the recording, measurement and burst interval (in conjunction with the decay)" on page 71.	
		Hold	Hold the last valid measurement value for x measurement cycles	
			off	Function deactivated
			1-5	Number of measurement cycles for which the measurement value is held until the error value is issued
			on	In the event of an error, the last valid measurement value is held until a new valid measurement value is present.

Universal inputs (0-20mA mode 2/2)	0-20mA (2/2)	Overflow	Procedure in the event of measurement range violations	
			Ignore	The measurement value is calculated beyond the range limits.
			Silent cutoff	The measurement value is truncated at the range limits.
			Overflow	• If the measurement value is above 20.1 mA, the error value "SC" (short circuit) is issued.
			NAMUR borders	• If the measurement value is above 20.1 mA, the error value "OF" (overflow) is issued. • If the measurement value is above 21 mA, the error value "SC" (short circuit) is issued.

Universal inputs (4-20mA mode 1/2)	4-20mA (1/2)	Filter time	Temporal averaging for signal smoothing	
		Decay	Temporal function in the measurement cycle	
			off	Decay deactivated
			min	The minimum of the last x measurement values is recorded.
			max	The maximum of the last x measurement values is recorded.
			avg	The arithmetic mean of the last x measurement values is recorded.
			med	The median of the last x measurement values is recorded.
			rms	The root mean square of the last x measurement values is recorded.
		Time	Time window for the decay. Up to a maximum of 60 measurement valuescan be used for the calculation (e.g. 1 channel: 60 measurement values; ... 4 channels: 15 measurement values each). To calculate the number of considered measurement values see "Example to clarify the recording, measurement and burst interval (in conjunction with the decay)" on page 71.	
		Hold	Hold the last valid measurement value for x measurement cycles	
			off	Function deactivated
			1-5	Number of measurement cycles for which the measurement value is held until the error value is issued
			on	In the event of an error, the last valid measurement value is held until a new valid measurement value is present.

Universal inputs (4-20mA mode 2/2)	4-20mA (2/2)	Overflow	Procedure in the event of measurement range violations	
			Ignore	The measurement value is calculated beyond the range limits.
			Silent cutoff	The measurement value is truncated at the range limits.
			Overflow	• If the measurement value is below 3.9 mA, the error value "OL" (open loop) is issued. • If the measurement value is above 20.1 mA, the error value "SC" (short circuit) is issued.
			NAMUR borders	• If the measurement value is below 3.6mA, the error value "OL" (open loop) is issued. • If the measurement value is below 3.9 mA, the error value "UF" (under flow) is issued. • If the measurement value is above 20.1 mA, the error value "OF" (overflow) is issued. • If the measurement value is above 21 mA, the error value "SC" (short circuit) is issued.

Universal inputs (0-2V mode 1/2)	0-2V (1/2)	Filter time	Temporal averaging for signal smoothing	
		Decay	Temporal function in the measurement cycle	
			off	Decay deactivated
			min	The minimum of the last x measurement values is recorded.
			max	The maximum of the last x measurement values is recorded.
			avg	The arithmetic mean of the last x measurement values is recorded.
			med	The median of the last x measurement values is recorded.
			rms	The root mean square of the last x measurement values is recorded.
		Time	Time window for the decay. Up to a maximum of 60 measurement valuescan be used for the calculation (e.g. 1 channel: 60 measurement values; ... 4 channels: 15 measurement values each). To calculate the number of considered measurement values see "Example to clarify the recording, measurement and burst interval (in conjunction with the decay)" on page 71.	
		Hold	Hold the last valid measurement value for x measurement cycles	
			off	Function deactivated
			1-5	Number of measurement cycles for which the measurement value is held until the error value is issued
			on	In the event of an error, the last valid measurement value is held until a new valid measurement value is present.

Universal inputs (0-2V mode 2/2)	0-2V (2/2)	Overflow	Procedure in the event of measurement range violations	
			Ignore	The measurement value is calculated beyond the range limits.
			Silent cutoff	The measurement value is truncated at the range limits.
			Overflow	If the measurement value is above 2.01 V, the error value "OF" (overflow) is issued.
			NAMUR borders	If the measurement value is above 2.01 V, the error value "OF" (overflow) is issued.

Universal inputs (0-10V mode 1/2)	0-10V (1/2)	Filter time	Temporal averaging for signal smoothing	
		Decay	Temporal function in the measurement cycle	
			off	Decay deactivated
			min	The minimum of the last x measurement values is recorded.
			max	The maximum of the last x measurement values is recorded.
			avg	The arithmetic mean of the last x measurement values is recorded.
			med	The median of the last x measurement values is recorded.
			rms	The root mean square of the last x measurement values is recorded.
		Time	Time window for the decay. Up to a maximum of 60 measurement valuescan be used for the calculation (e.g. 1 channel: 60 measurement values; ... 4 channels: 15 measurement values each). To calculate the number of considered measurement values see "Example to clarify the recording, measurement and burst interval (in conjunction with the decay)" on page 71.	
		Hold	Hold the last valid measurement value for x measurement cycles	
			off	Function deactivated
			1-5	Number of measurement cycles for which the measurement value is held until the error value is issued
			on	In the event of an error, the last valid measurement value is held until a new valid measurement value is present.

Universal inputs (0-10V mode 2/2)	0-10V (2/2)	Overflow	Procedure in the event of measurement range violations	
			Ignore	The measurement value is calculated beyond the range limits.
			Silent cutoff	The measurement value is truncated at the range limits.
			Overflow	If the measurement value is above 10.05 V, the error value "OF" (overflow) is issued.
			NAMUR borders	If the measurement value is above 10.05 V, the error value "OF" (overflow) is issued.
Ext. temperature sensor	off	---	Measurement channel deactivated	
	on	Hold	Hold the last valid measurement value for x measurement cycles	
			off	Function deactivated
			1-5	Number of measurement cycles for which the measurement value is held until the error value is issued
			on	In the event of an error, the last valid measurement value is held until a new valid measurement value is present.

9.2.1.4.3 Alarms

"Digital" mode	WA	A "high" at the universal input triggers a "warning".	
	AL	A "high" at the universal input triggers an "alarm".	
	SW	A "high" at the universal input triggers a "fault warning".	
	SA	A "high" at the universal input triggers a "fault alarm".	
All other modes	Warning	Value low	A warning is triggered, if the measurement value drops to or below this value.
		Value high	A warning is triggered, if the measurement value meets or exceeds this value.
	Alarm	Value low	An alarm is triggered, if the measurement value drops to or below this value.
		Value high	An alarm is triggered, 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)	

9.2.1.4.4 Trigger

"Digital" mode	QU	Fastrecording (recording interval = recording interval / factor)	
	SL	Slow recording (recording interval = recording interval * factor)	
	MS	Start measurement cycle immediately	
	XM	Initiate transmission	
	ON	Activate online mode	
	I1	Internal use with the control program	
	I2	Internal use with the control program	
	I3	Internal use with the control program	
	I4	Internal use with the control program	
	Edges	Selection of the edge at which the trigger should be initiated	
		rising	Rising edge initiates the trigger.
		falling	Falling edge initiates the trigger.
		both	Both edges initiate the trigger.
All other modes	QU	Fastrecording (recording interval = recording interval / factor)	
	SL	Slow recording (recording interval = recording interval * factor)	
	RO	Switch on recording	
	RF	Switch off recording	
	XM	Initiate transmission	
	ON	Activate online mode	
	I1	Internal use with the control program	
	I2	Internal use with the control program	
	I3	Internal use with the control program	
	I4	Internal use with the control program	
	Level	Levels for initiating the trigger	
		greater or equal	The measurement value must be greater or equal to the level to initiate the trigger.
		less or equal	The measurement value must be less or equal to the level to initiate the trigger.

9.2.1.5 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.*

9.2.1.5.1 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 that is used as a 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 that is used as a 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

9.2.1.5.2 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

9.2.1.5.3 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 triggered, if the measurement value drops to or below this value.
Alarm high	An alarm is triggered, 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)

9.2.1.6 Output channels

9.2.1.6.1 Basis

Ext warmup time

Indicates the amount of time that an output channel is switched on in "Ext warmup time" mode before the measurement.

Title 1

Freely selectable channel title for the isolated switch contact

Title 2

Freely selectable channel title for the analogue output (no galvanic separation)

Mode

Basic setting for output channel 1 (isolated switch contact)

Off	*---*	*Deactivated output channel*
Ext warmup time	*The output channel is switched on "ext warmup time" seconds prior to the measurement. If the value is "0", the output channel is not switched on at all.*	
Digital	*Invert*	*Inverts the level issued on the device (see "Supplementary explanation of "Digital" mode" on page 67).*
	Setpoint	*Setpoint (on/off) that should be issued*
Freq	*Factor*	*The setpoint multiplied by the factor equals the frequency in Hertz.*
	Min	*Defines the lower scale end of the pointer instruments*
	Max	*Defines the upper scale end of the pointer instruments*
	Setpoint	*The setpoint multiplied by the factor equals the frequency in Hertz.*
	Unit	*String that is used as a 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*
PWM	*0%*	*Start of the measurement range in the measuring unit*
	100%	*End of the measurement range in the measuring unit*
	Setpoint	*Output value in the measuring unit*
	Unit	*String that is used as a 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*
Impulse	*Factor*	*The setpoint multiplied by the factor equals the impulse/min.*
	Min	*Defines the lower scale end of the pointer instruments*
	Max	*Defines the upper scale end of the pointer instruments*
	Setpoint	*The setpoint multiplied by the factor equals the impulse/min.*
	Unit	*String that is used as a 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*

Basic setting for output channel 2 (analogue output)

Off	*---*	*Deactivated output channel*
0-20mA	*0%*	*Start of the measurement range in the measuring unit*
	100%	*End of the measurement range in the measuring unit*
	Unit	*String that is used as a 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*
4-20mA	*0%*	*Start of the measurement range in the measuring unit*
	100%	*End of the measurement range in the measuring unit*
	Unit	*String that is used as a 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*

Note:

Supplementary explanation of "Digital" mode

Invert	*Setpoint*		*Output on the device*
Off	*Off*	=	*Off (low)*
Off	*On*	=	*On (high)*
On	*Off*	=	*On (high)*
On	*On*	=	*Off (low)*

9.2.1.6.2 Config

Off	---	
Ext warmup time	---	
Digital	---	
Freq	---	
PWM	---	
Impulse	Duration	Duration of an output impulse
0-20mA	---	
4-20mA	---	

9.2.1.7 Internal channels

9.2.1.7.1 Basis

Voltage title	Freely selectable channel title for the supply voltage	
	Unit	String that is used as a measurement unit by all of the server display elements
GSM level title	Freely selectable channel title for the GSM field level	
	Unit	String that is used as a measurement unit by all of the server display elements
Int. temperature title	Freely selectable channel title for the internal device temperature	
	Unit	Selection of the temperature unit used by all of the server display elements

9.2.1.7.2 Alarms

Warning	Value low	A warning is triggered, if the measurement value drops to or below this value.
	Value high	A warning is triggered, if the measurement value meets or exceeds this value.
Alarm	Value low	An alarm is triggered, if the measurement value drops to or below this value.
	Value high	An alarm is triggered, 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)	

9.2.1.7.3 Trigger

QU	Fastrecording (recording interval = recording interval / factor)	
SL	Slow recording (recording interval = recording interval * factor)	
RO	Switch on recording	
RF	Switch off recording	
XM	Initiate transmission	
ON	Activate online mode	
I1	Internal use with the control program	
I2	Internal use with the control program	
I3	Internal use with the control program	
I4	Internal use with the control program	
Level	Levels for initiating the trigger	
	Greater or equal	The measurement value must be greater or equal to the level to initiate the trigger.
	Less or equal	The measurement value must be less or equal to the level to initiate the trigger.

9.2.1.8 Alarm settings

Acknowledgement	Standard	The global server setting is used (see "myDatanet Server 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 "myDatanet Server 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.	

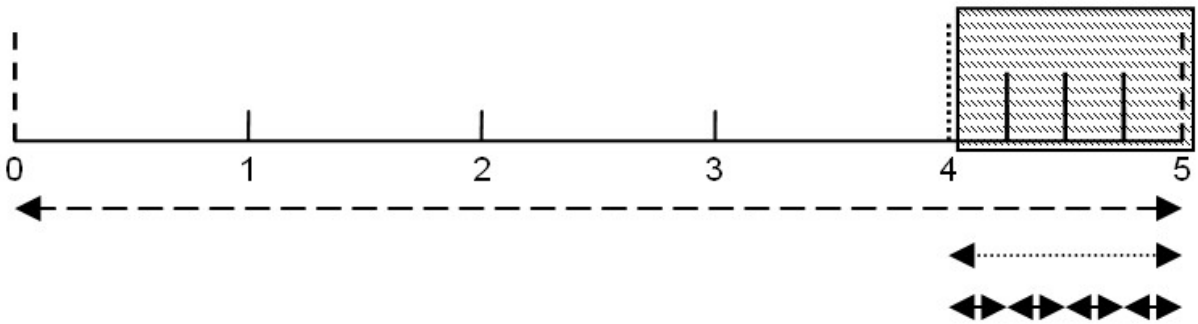
9.2.1.9 Basic settings

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 "myDatanet Server 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 to clarify the recording, measurement and burst interval (in conjunction with the decay)

Basic setting		*Record interval*	*5 min.*
		Burst interval	*1 min.*
		Measurement cycle	*15 sec.*
Measurement channels - ›Config.		*Decay*	*med*
		time	*60 sec.*



Formation of the measurement value: The last four measurement values are used to determine the median and recorded as the measurement value.

Note: *Additional explanation of the connection types*

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

9.2.1.10 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 "myDatanet Server 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	

9.2.2 Device configuration

> ***Note:*** *Several of the configuration fields in the following sub chapters may possibly be hidden depending on the respective user level. In this case, 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 "myDatanet Server Manual " 206.886) or by clicking on the device name in the device name list (see "myDatanet Server Manual " 206.886).

9.2.2.1 Comments

Comments

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

9.2.2.2 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).

9.2.2.3 Device-specific settings

Operating type	HOLD	Measurement: OFF, transmission: ON
	RUN	Measurement: ON, transmission: ON
	OFF	Measurement: OFF, transmission: OFF
	OFFLINE	Measurement: ON, transmission: OFF
	Aloha	Reserved for internal use

9.2.2.4 GPRS

SIM tariff

Selected SIM tariff

Chapter 10 Instruction list

10.1 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 myDatalogS3 :

- 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 entry of an instruction list

The instruction list is entered via the "Control" configuration section (see "Control" on page 46) of the input screen for configuring the site. "IL" must be selected as the "script type" so that myDatalogS3 interprets the commands entered in the "script" as an instruction list.

10.1.2 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 46) of the input screen for configuring the site (to create a template, see "myDatanet Server 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 myDatalogS3 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.*

10.2 Implementation details

10.2.1 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

10.2.2 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 %ID0 LT 10 JMPCN label1 ADD 3 label1: SUB 2

10.3 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 147).

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 78 and "Expansions" on page 82)
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 76, 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 76 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 76 and "Expansions" on page 82 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

10.4 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)]myDatalogS3 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

10.5 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 .. %Ix8 • %Ix0...%Ix2: Universal inputs 1-3 • %Ix3: Internal measurement channel "GSM level" • %Ix4: Internal measurement channel "voltage" • %Ix5: Internal measurement channel "int. temp" temp" • %Ix6: ext. temperature sensor • %Ix7 to %Ix8: Output channel 1-2
Output addresses	%Qx0 .. %Qx1 • %Qx0 to %Qx1: Output channel 1-2
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: I1-I4
	  • Bit8: Quick recording
	  • Bit9: Slow recording
	  • Bit10: Start measurement cycle immediately
	  • 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: Warmup time active
	  • Bit17: Measurement active
	  • Bit18-19: Reserved
	• %Mx1001: Current time stamp

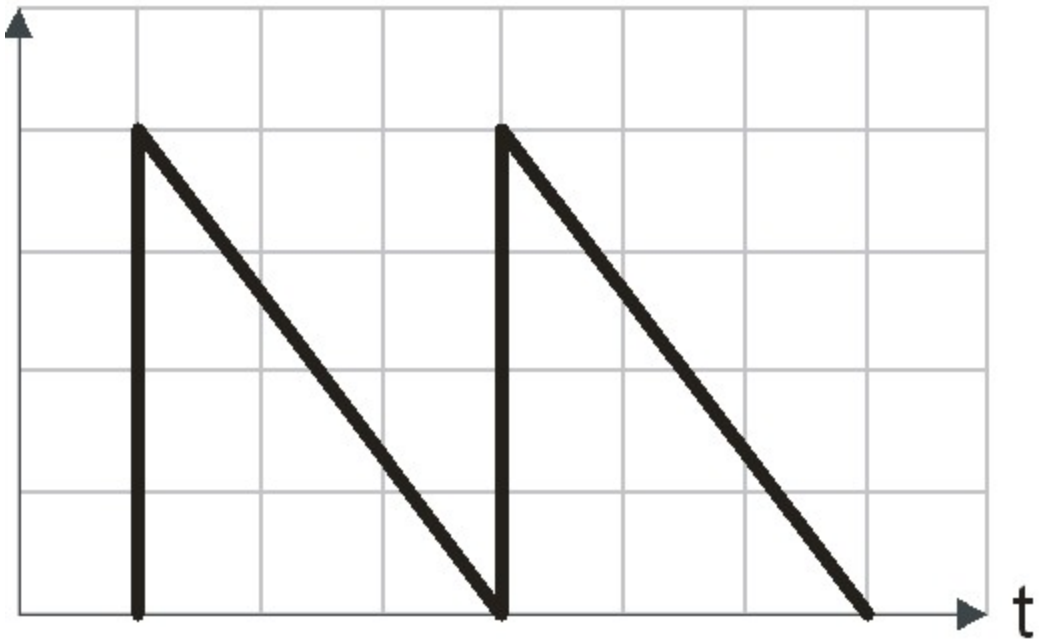
Register addresses that specify which triggers are active or inactive for the relevant input.	%Mx2000 .. %Mx2008 • Bit0-3: Reserved • Bit4-7: I1-I4 • Bit8: Quick recording • Bit9: Slow recording • Bit10: Start measurement cycle immediately • Bit11: Switch on recording • Bit12: Switch off recording • 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 .. %Mx3008 • Bit0: Warning • Bit1: Alarm • Bit2: Fault warning • Bit3: Fault alarm
Display register addresses	%Mx4000 .. %Mx4007

10.6 Example

10.6.1 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 *)
```

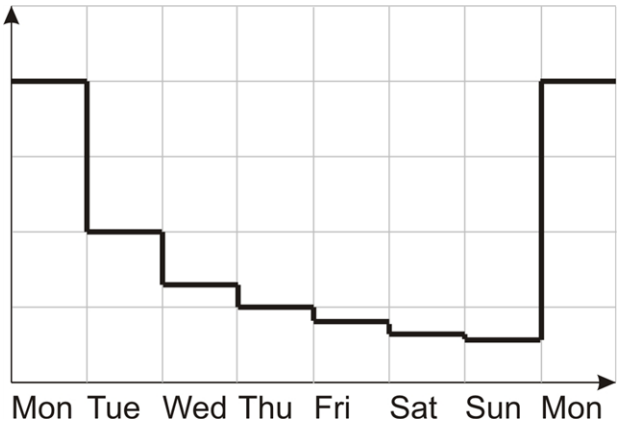


Saw-tooth generated by means of IL

10.6.2 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 *)
```

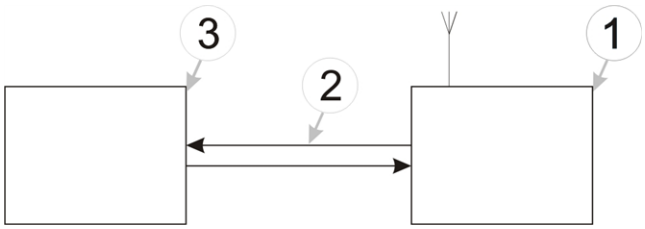


Output signal generated by means of IL

10.6.3 Using the display registers

10.6.3.1 Task definition

The values measured on the universal inputs must be issued via the RS232. If the value on the universal input is invalid (NaN), the value issued by the RS232 must be set to zero.



Schematic diagram

1 myDatalogS3	3 Device to display the data, e.g. a PC
2 RS232 interface	

10.6.3.2 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 example will not be provided.

Control	Script type	IL
	Script	see "Instruction list" on page 89

10.6.3.3 Instruction list

```
  LDS %IR0                    (* Loads the status of universal input 1 into the accu *)
  JGTZ Channel1_invalid       (* Measurement value invalid -> jump to the "Channel1_invalid" label *)
  LD %IR0                     (* Loads the current measurement value of universal input 1 into the accu *)
  ST %MR4000                  (* Write the value from the accu in display register 0 *)
  JMP channel2                (* Jump to the "Channel2" label *)

Channel1_invalid:             (* "Channel1_invalid" label (Measurement value invalid) *)
  LD 0.0                      (* Load 0 into the accu *)
  ST %MR4000                  (* Write the value from the accu in display register 0 *)

Channel2:                     (* "Channel2" label *)
   LDS %IR1                   (* Load the status of universal input 2 into the accu *)
   JGTZ Channel2_invalid      (* Measurement value invalid -> jump to the "Channel2_invalid" label *)
   LD %IR1                    (* Load the current measurement value of universal input 2 into the accu *)
   ST %MR4001                 (* Write the value from the accu into display register 1 *)
   JMP channel3               (* Jump to the "Channel3" label *)

Channel2_invalid:             (* "Channel2_invalid" label (measurement value invalid) *)
  LD 0.0                      (* Load 0 into the accu *)
  ST %MR4001                  (* Write the value from the accu into display register 1 *)

Channel3:                     (* "Channel3" label *)
   LDS %IR2                   (* Loads the status of universal input 3 into the accu *)
   JGTZ Channel3_invalid      (* Measurement value invalid -> jump to the "Channel3_invalid" label *)
   LD %IR2                    (* Loads the current measurement value of universal input 3 into the accu *)
   ST %MR4002                 (* Write the value from the accu into display register 2 *)
   JMP End                    (* Jump to the "End" label *)

Channel3_invalid:             (* "Channel3_invalid" label (measurement value invalid) *)
  LD 0.0                      (* Load 0 into the accu *)
  ST %MR4002                  (* Write the value from the accu into display register 1 *)

End:                          (* "End" label *)
```

10.6.3.4 Communication on the interface

```
+DISP: 12/06/28 17:23:10;0.00000000;5.39930868;0.00000000;

+DISP: 12/06/28 17:23:20;0.00000000;5.39930868;0.00000000;

+DISP: 12/06/28 17:23:30;0.00000000;5.39930868;0.00000000;

+DISP: 12/06/28 17:23:40;0.00000000;5.39930868;0.00000000;
```

Explanation

+DISP: 12/06/28 17:23:10;0.00000000;5.39930868;0.00000000;

Shows the start of the display register string

+DISP: **12/06/28 17:23**:10;0.00000000;5.39930868;0.00000000;

Time stamp of the display register string

+DISP: 12/06/28 17:23:10;**0.00000000**;5.39930868;0.00000000;

Display register 0

+DISP: 12/06/28 17:23:10;0.00000000;**5.39930868**;0.00000000;

Display register 1

+DISP: 12/06/28 17:23:10;0.00000000;5.39930868;**0.00000000**;

Display register 2

Chapter 11 Pawn script

11.1 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 myDatalogS3 :

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

11.1.1 Direct input of a pawn script

The pawn script is entered via the "Control" configuration section (see "Control" on page 46) of the input screen for configuring the site. "Pawn" must be selected as the "script type" so that myDatalogS3 interprets the commands entered in the "script" as a pawn script.

11.1.2 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 46) of the input screen for configuring the site (to create a template, see "myDatanet Server 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 myDatalogS3 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.*

11.2 General syntax

11.2.1 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.

11.2.2 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.

11.2.3 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.

11.2.4 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.

11.2.5 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			

11.2.6 Numerical constants

11.2.6.1 Numerical integer constants

Binary:

0b followed by a series of 0 and 1

Decimal:
A series of numbers between 0 and 9

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

11.2.6.2 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.

11.3 Variables

11.3.1 Declaration

The keyword "new" declares a new variable. For special declarations, the keyword "new" is replaced with "static" (see "Static local declaration" on page 94). 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 105) - local variable

Example:

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

11.3.2 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.

11.3.3 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.

11.3.4 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.

11.3.5 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.

11.3.6 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)
```

11.4 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 110.

11.5 Array variables

11.5.1 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]".

11.5.2 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 */
```

11.5.3 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
```

11.5.4 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 96).

11.5.5 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.

11.6 Operators and expressions

11.6.1 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*

11.6.2 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
```

11.6.3 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".*

11.6.4 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.

11.6.5 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

11.6.6 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.

11.6.7 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".

11.6.8 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.

11.6.9 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

11.7 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.

11.7.1 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.

11.7.2 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".

11.7.3 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.

11.7.4 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.

11.7.5 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
}
```

11.7.6 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
}
```

11.7.7 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
}
```

11.7.8 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
}
```

11.7.9 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.

11.7.10 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.

11.7.11 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).

11.7.12 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
}
```

11.7.13 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.

11.7.14 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
}
```

11.7.15 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
}
```

11.8 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 97.

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.

11.8.1 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 */
```

11.8.2 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.

11.8.3 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]
}
```

11.9 Extensions

11.9.1 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,
};
```

11.9.2 Device-dependent channel numbers

Definition of the constants	Explanation
MDN_CH_UI1 : MDN_CH_UI3	Universal inputs 1-3
MDN_CH_GSM	Internal measurement channel "GSM level"
MDN_CH_VIN	Internal measurement channel "voltage"
MDN_CH_TEMP	Internal measurement channel "Int. temperature"
MDN_CH_1WTEMP	ext. temperature sensor
MDN_CH_DIGOUT	Digital output
MDN_CH_MAOUT	Analogue output
MDN_NUM_CHANNELS	Total number of all channels (all of the in- and outputs)

11.9.3 Functions

11.9.3.1 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"*

11.9.3.2 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*

11.9.3.3 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 115)*
value	*Value of the channel*
status	*Status of the channel (see "Status of the measurement value channel" in the chapter "Constants" on page 114) - 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 115)*
value	*Value of the channel*
status	*Status of the channel (see "Status of the measurement value channel" in the chapter "Constants" on page 114) - 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 115)*

	Explanation
Return value	*Positive, if successful (see "Alarm flags" in the chapter "Constants" on page 114), 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 115)*
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 114).*

	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.*

11.9.3.4 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.

11.9.4 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 147).

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.

11.10 Example

11.10.1 Saw-tooth generator

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

```
#include <mdlMINI>                          /* Load the definitions for the
                                            myDatalogS3  */

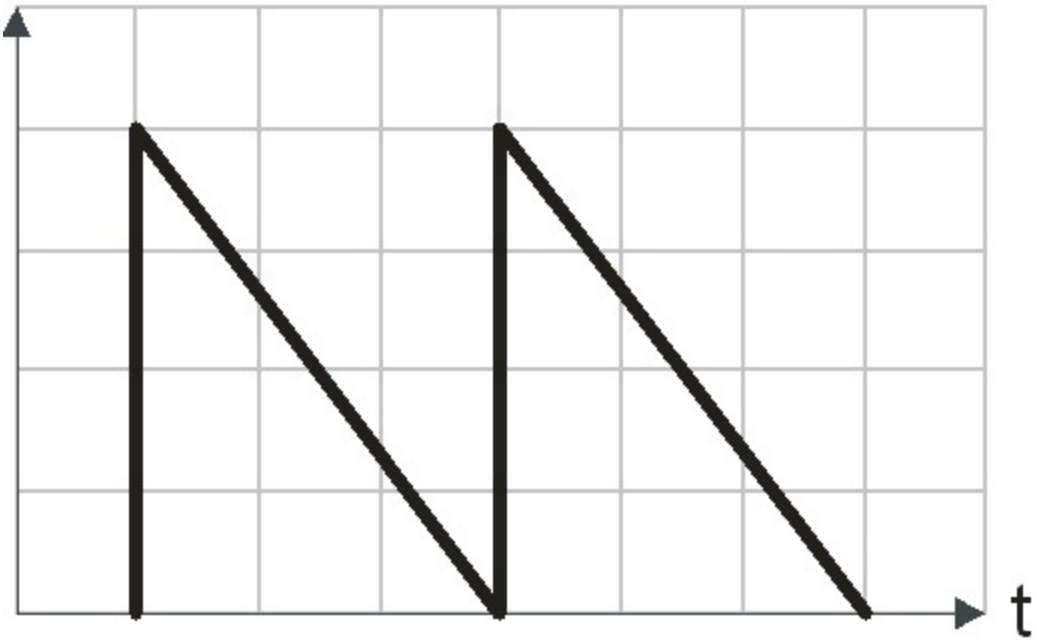

forward public Mdn_CtrlFinish();            /* Declaration of the function that is called up at the end of the
                                               measurement cycle. */

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

11.10.2 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 <mdlMINI>                                  /* Load the definitions for the
                                                    myDatalogS3  */


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. */

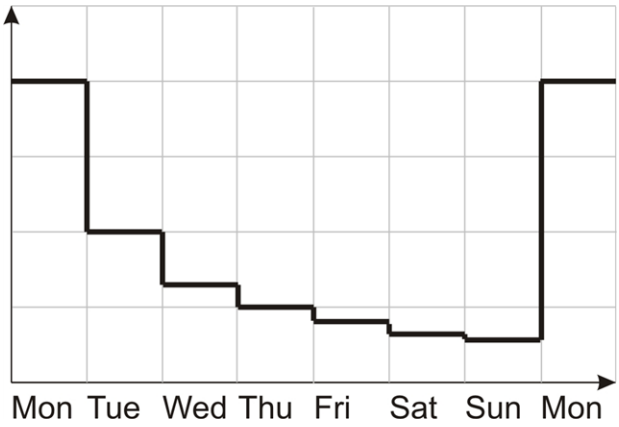
    iDayOfWeek = Mdn_DoW(Mdn_GetDate());            /* Save weekday to the iDayOfWeek
                                                       variable */
    iDayOfWeek += 1;                                /* +1 so that the subsequent division is not
                                                      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

11.10.3 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 <mdlMINI>                                  /* Load the definitions for the
                                                    myDatalogS3 */


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
}
```

11.11 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 92).
- 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 110). The "placeholder" argument replaces some applications of the ZERO pointer (see "Standard values of function arguments" on page 112).
- Number can be specified in a hexadecimal, decimal or binary format. The octal format is not supported (see "Numerical constants" on page 92). 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 107). 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 101.
- An empty statement is an empty block (with {}) and not a semicolon (see "Composite statements" on page 103). 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 97).

- 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 12 Maintenance

Important note: *The work described in this section of these instructions may only be performed by qualified personnel to prevent damage to the instrument.*

Before maintenance, cleaning and/or repair work, the device must absolutely be deenergised (zero potential).

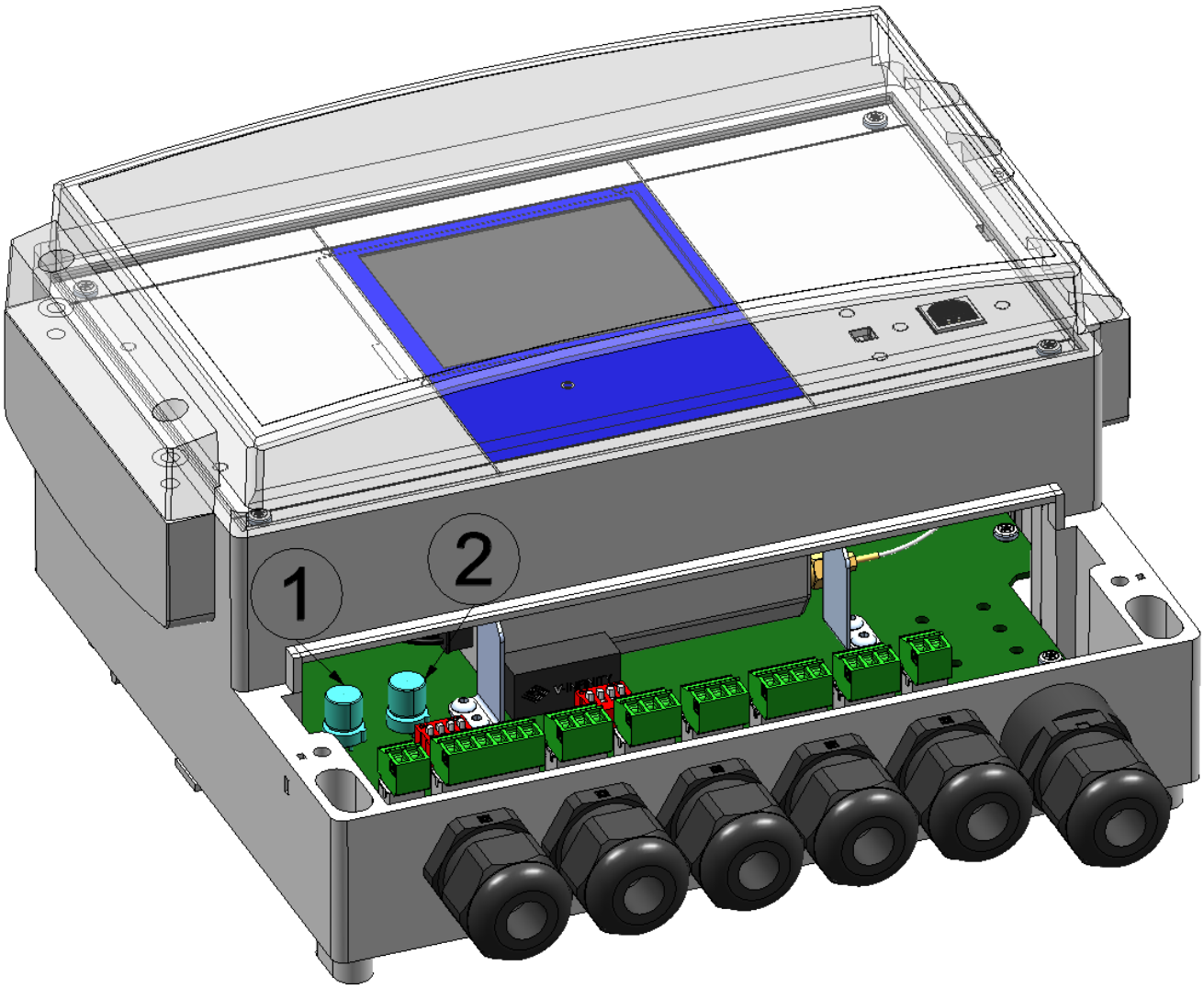
12.1 General maintenance

- Regularly check the myDatalogS3 for mechanical damage.
- Check all of the connections for leaks or corrosion on a regular basis.
- Check all of the cables for mechanical damage at regular intervals.
- Clean the myDatalogS3 with a soft, moist cloth. Use a mild cleaning agent, if necessary.

12.2 Replacing the fuse



DANGER:
Risk of fire. An incorrect fuse can cause injuries, damages or emissions. Only replace fuses with the same type and specification of fuses.



Replacing the fuse

1 T2A 250V fuse (206.312)	**2** Replacement fuse

1. First of all disconnect the power supply, before removing both of the screws that secure the housing compartment cover. Then open the housing compartment of the myDatalogS3 .

   Important note: *In the event of adverse weather conditions including rain or in a location where water can penetrate from above, suitable measures must be implemented to protect the device from penetrating moisture when the housing compartment cover is open.*

2. Remove the fuse (1) from the socket.
3. Insert a new T2A 250V fuse.

   Note: *The myDatalogS3 is supplied with a replacement fuse that is located in its own slot (2) directly next to the actual fuse.*

4. Close the housing compartment cover.

   Important note: *Ensure that the seals are clean and intact before closing the housing compartment cover. Remove any impurities and/or dirt. The manufacturer shall not be liable for any damage to the device caused by leaky or faulty seals.*

5. Switch on the 24 VDC power supply of the myDatalogS3 . The status LED should then start to flicker (see "Status LED" on page 44), indicating that a connection is being established.

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 40).

6872
6872
6872

Chapter 13 Removal/disposal

Dispose of the device in accordance with the locally valid environmental regulations for electronic products.

6872
6872
6872

Chapter 14 Troubleshooting and repair

14.1 General problems

Problem	Cause/solution
Device does not respond (status LED always off)	• Check the cable connections (see "Connecting the sensors, actuators and power supply" on page 31) • Check the fuse (see "Replacing the fuse" on page 133)
Communication problems	• Evaluate the status LED blink code (see "Status LED" on page 44). • Load the device log from the myDatalogS3 or the myDataweb server and use myDatanetDeviceConfig for the evaluation (see "Evaluating the device log" on page 147).
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 "myDatanet Server Manual " 206.886). Solution: Initiate a ALOHA transmission or wait for the cyclical transfer. • The assignment of the device and site is not correct (see "Site" on page 45).
Data at universal input is not plausible.	• Check whether the universal input configuration matches the sensor output signal (see "Measurement channels" on page 46). • Check the filter settings of the universal input (see "Measurement channels" on page 46).
An analogue channel alarm status has not been identified.	Increase the measurement cycle
Alarm status 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 "myDatanet Server Manual " 206.886). Solution: Initiate a ALOHA transmission or wait for the cyclical transfer.
Alarm message was not sent although the alarm was transmitted to the myDataweb server.	• Check the settings of the alarm call plan (see "myDatanet Server Manual " 206.886). • Check the address data of the alarm call plan (see "myDatanet Server Manual " 206.886).
Isolated switch contact is not working	• Disruption to the voltage that is conducted via the isolated switch contact • The site settings were overwritten by a server PLC.

Problem	Cause/solution
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 46). • Load the device log from the myDataweb server and use myDatanetDeviceConfig for the evaluation (see "Evaluating the device log" on page 147). 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 80).
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 46). • Load the device log from the myDatawebserver and use myDatanetDeviceConfig for the evaluation (see "Evaluating the device log" on page 147). 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 125).

14.2 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 153).
1200	MODEM ERROR			Modem error (see "Modem error" on page 144)
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 153).
1282	ZLIB STREAMFINISH ERR	##	---	Internal error Contact the manufacturer if the device log includes the error several times (see "Contact information" on page 153).
1440	DECAY MEM ERR	---	---	Too many measurement values are being taken into consideration during the decay (see "Measurement channels" on page 46). Either reduce the time slot for the decay or the measurement cycle.

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 myDatalogS3 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 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			reserved for extensions
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 80 • "Pawn script error codes" on page 125

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

14.2.1 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.

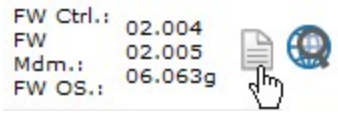
Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1200	BAND SEL FAILED	-969	---	A network could not be found on the GSM900/1800 or on the GSM85071900 band. • Try to improve the position of the antenna. • Check whether the device is in the coverage area (www.microtronics.at/footprint).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

14.3 Evaluating the device log

14.3.1 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 ("myDatanet Server Manual " 206.886) includes a detailed description of the evaluation of the device log on the myDataweb server.

14.3.2 Evaluating the device log using myDatanetDeviceConfig

The myDatanetDeviceConfig program can be used to read all of the stored log entries, including those that have not yet been transferred to the myDataweb server, directly from the myDatalogS3 via the USB interface.

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

Chapter 15 Spare parts and accessories

15.1 Internal expansions

Note: The following components are order options. They cannot be ordered separately or installed by the customer.

Description	Quantity	Order number
myDatalogS3 230VAC Option todo	1	206.680

15.2 Antennas

Description	Quantity	Order number
Antenna 900 FME	1	206.801
Flat antenna Smart Disc FME-F	1	206.814
Extension cable for Antenna 900 FME	1	206.805
ANT OutSide GP	1	206.607
Flat Antenna Smart Disc US FME-F	1	206.817
Antenna 1800 FME	1	206.809
Antenna adapter FME-F to SMA-F	1	206.822
Flat antenna Smart Disc Plus FME-F	1	206.824
Antenna extension SMA-F to FME-F	1	206.825
Flat whip antenna with adapter SMA-FME	1	206.830
Portable Antenne Multi Band	1	206.826

15.3 Sensors

Description	Quantity	Order number
Temperatur sensor TEMP1	1	269-TEMP1

15.4 Other accessories

Description	Quantity	Order number
myDatanet Tool Pen	1	206.646
MDN Magnet	1	206.803
Terminal box	1	206.622
NG24	1	213.814
USB A/B Kabel	1	206.601
Replacement fuse MDL8	1	206.312

Chapter 16 Document history

Rev.	Date	Changes
02	18.12.2012	**Chapter "Selection of the GSM network" on page 24** *Added chapter describing the operation of the GSM network selection* **Chapter "Script eradication" on page 25** *Chapter added* **Chapter "Control" on page 46** *Selection of the pawn script* *Explanation adjusted to the modified input screen* **Chapter "Device-specific implementation" on page 84** *Error in specifying the output addresses corrected.* **Chapter "Pawn script" on page 91** *Added chapter describing the functionality of the pawn script* **Chapter "Troubleshooting and repair" on page 139** *Error descriptions and solutions supplemented* **Chapter "Log entries and error codes " on page 141** *Breakdown of the error codes added*

6872
6872
6872

Chapter 17 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