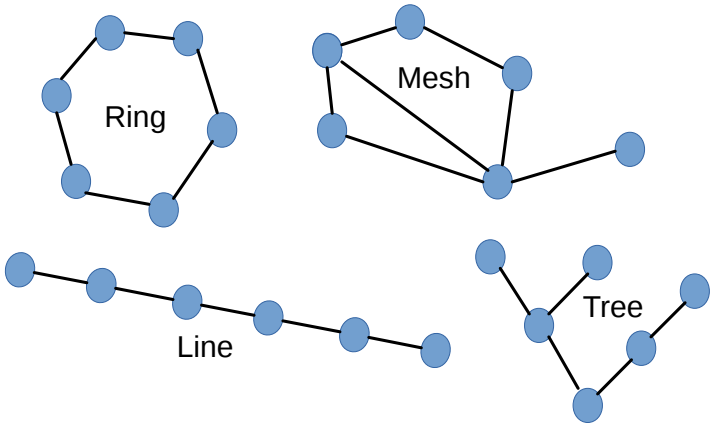
Link Network Protocol via RS-485

Specification and implementation guide

Version 0.10.1, 2017-11-21

Author: Peter Seng



intentionally blank

Please read carefully:

Copyright

PUBLISHED BY: SENG DIGITALE SYSTEME, PETER SENG
ALEXANDERSTRASSE 14, 73054 EISLINGEN, GERMANY

Disclaimer of Warranty

Limitation of Liability

Contents

1 About

this guide describes the specification and implementation of a peer-to-peer messaging network communication protocol, based on point-to-point RS-485 transmission lines connecting nodes called "*Link Network Protocol via RS-485"*.

In sense of this document a *node* is an active element connecting transmission lines, *devices* are active elements containing node functionality and running an application program. A *link* is a transmission line connecting two nodes and a *link network* is a network based on links and nodes. Data frames transported via links are called *messages*.

Every node has a minimum of two RS-485 *link interfaces*, so that basic networks can be arranged in line or ring topology. Networks including nodes with more than two link interfaces can build arrays in tree or meshed topology.

Devices integrating "*Link Network Protocol via RS-485*" are designed to comply with "*Smart Devices Messages Network"* (SDMN) requirements. SDMN devices imply extended identification features and unified message content to ease communication between devices attached to or connected by heterogeneous links or networks.

Cross-references within this document a formated like this.

Any comments concerning this guide are welcome, please mail to: info@seng.de

2 Features

Like every network or bus system this approach has it's features that have to match the application it will be used in:

- message based protocol including priority
- sea of nodes architecture, all nodes have equal rights
- no client/server architecture, no tokens, no polling of node addresses
- based on stable and rugged RS-485 transmission lines
- reliable messages transport due to embedded CRC16 checksum
- install and run without hassle about addressing of nodes or cable termination
- easy localization of defect node or transmission line
- link status visualization via LED presenting link traffic and errors
- breakdown of a node can not disturb the complete network. At worst case, if network topology is a line or tree it will lead to a segmentation of the network
- realization of fault tolerant networks based on meshed or ring topology is possible
- every node includes unique 48-bit ID, pre-programmed during manufacturing
- addressing of single devices, groups and device-types implemented
- automatic detection and resolution of address conflicts doable
- designed to be comply with SDMN requirements
- all devices include repeater functionality
- at default bit rate of 115200 bit/s distance between nodes can be up to 1000 m
- data rate and cable length combinations can vary from 100 kbit/s at 1200 m to 10 Mbit/s at 5 m
- in one network links can run at different speed
- no special or proprietary semiconductors needed
- inexpensive twisted pair cabling
- most times compatible to already installed cabling
- build in USB network access point for configuration and data exchange at property U devices
- configure network and devices by use of a terminal program
- no need for special configuration and programming software
- realize fiber optic cabling or galvanic isolation by using off-the-shelf converters
- message consists of 0...48 bytes payload and a constant overhead of 16 bytes, resulting in a message length of 16...64 bytes
- message transmission time at max. message length is less than 10 ms between nodes.
- enables the design of plug and play applications

3 Applications

- Industrial-, building-, home-automation and control
- Sensor and actor communication
- HVAC control
- Access control
- …..

4 Implementation challenges

To achieve all the features mentioned some challenges had to be resolved:

- in order to address nodes these must have a unique identifier
- simultaneous serial transmission of messages on multiple ports represents a high load for the processing unit controlling the node
- the RS-485 standard is known to be the base of reliable systems, but there exist two intrinsic difficulties:
  - transmission lines have to be terminated, leading to problems during installation and operation
  - in multi-master systems the allocation of the transmission line must be regulated to avoid or detect collisions on the line
- allocation of transmission lines and message spreading / routing must be resolved
- in ring or meshed networks the same message can exist (and so be received) multiple times due it is transmitted on different paths
- system behavior on open links must be resolved
- configuration of the system must be resolved

Some of features might have drawbacks that have to be discussed, solved and rated. Evaluating the result will enable the designer to make the decision of integrating the
"*Link network protocol via RS-485"* into a special application, implement modifications or to chose another better matching communication standard.

5 SDMN - node identification

For identification of SDMN network nodes some rules have to be applied.

5.1 Manufacturer-identifier

A manufacturer intending to produce SDMN devices has to register for a **M**anufacturer-**ID**entifier (MID). The MID is a unique 24 bit number, programmed into the device during the manufacturing process.

For companies already registered for a MA-L (previously named OUI) identifier, assigned by IEEE (http://www.ieee.org), and whose ID is less than 0xFF0000 it is mandatory to use the value of the MA-L for MID.

In all other cases a handout of a MID and it's registration can be requested from the author of this document by payment of a nominal charge. Registration and publication is reached by entry of manufacturer name, address and MID into the appendix "18.2 SDMN - MID table" of this document.

Registration and public listing of the MID is mandatory to ensure compatibility of SDMN devices from different manufacturers attached to the same network.

The use of an unregistered MID assumes the ability of the operator to select an MID not interfering with MID based MAC's on the same network. It is strictly recommended to limit the use of an unregistered MID to private and evaluation use only.

5.2 Type-identifier

A manufacturer producing SDMN devices has to assign a unique number to every product type.

Devices that are not fully compatible in hard- and firmware have to carry different type-identifiers.

Type-identifier is a 24-bit number, minimum value is 0x000001.

5.3 Device-type

The device-type is programmed into the device during a firmware update procedure. It should be hard coded, defined by firmware to prevent modification. The device-type is 48 bits long, organized in 2 groups:

- upper 24-bits are defined by the MID (see 5.1)
- lower 24-bits identify the type-identifier (see 5.2)

The device-type is structured according to following base16 format: “003C7E 0007CA”.
One use of the identifier is to address compatible devices inside a network for the transfer of firmware updates. Another use is to identify devices in a unknown network or to verify the documentation of the network structure. See “15.1.4 AdrType” for enabling this kind of addressing. A read out of the device-type via the network must be implemented.
Devices that are not fully compatible in hard- and firmware have to carry different device-types.

Erroneous device-type assignment may lead to none trivial system errors and a damage of the devices during firmware update procedures.

5.4 Serial number

SDMN devices carry a unique 48-bit **S**erial **N**umbe**R** (SNR), unequal to zero. It is programmed into the device during the manufacturing process. A SNR is a unique number, identifying one specific device produced by a manufacturer, the **same number can only be assigned once to any physical device** produced by the manufacturer. SNR must not change during lifetime of the product nor assigned to any other device of the same manufacturer after lifetime.

So when different types of SDMN devices are produced by one manufacturer:

1. the assignment of serial numbers has to be synchronized (favored, see remark)

or

2. the 48-bit number range has to be segmented by using some higher bits for identifying the type and the lower bits for the serial number. The partition of these 48-bits is adaptable to manufacturers requirements (unfavored, see remark)

Remark: Cause only the 24 low order SNR bits are represented by the MAC (see 5.5), segmentation leads to tremendous higher probability of initial MAC address conflicts inside a network. Although these can be automatically solved, avoiding conflicts is preferable. The higher probability results from the effect that several types are produced in parallel within the (low order) 24-bit address range, so different types of SDMN devices from one manufacturer within a network may carry the same default MAC.

5.5 Media access code

Every SDMN device inside a network is identified via it's unique **M**edia-**A**ccess-**C**ode (MAC). The default MAC is programmed into the device during the manufacturing process. The MAC is 48-bits long, organized in 2 groups:

- upper 24-bits are for manufacturer identification (MID) and can not be altered by setup (example: “0x003C7E”)
- lower 24-bits, by default, represent the lower 24-bits of the SNR

The lower 24-bits of the MAC may be altered by device configuration:

1. to enable replacement of a (functionally compatible) device without the need to change system configuration
2. to keep MAC identifiers unique, in the rare condition, when lower 24-bits of SNR of devices from the same manufacturer (inside one network) are identical

The singularity of the MAC inside one network has to be ensured, double assignments of MAC's leads to none trivial system errors.

5.6 Identification label

For identification purposes each SDMN device carries a label. The label carries following information:

1. the string “SDMN”
2. the string “MID”, followed by the MID in base-16 format. Example: “MID 003C7E”.
3. the string “SNR” followed by the SNR, according to one of the following base-16 formats:
   - Single string format: “SNR 000000 001B2D”
   - Double string format: “SNL 001B2D”, “SNH 000000”
   - Serial numbers with the upper 24-bits all zero can be denoted in the form: “SNR 001B2D”

If changed by device configuration the content of the MAC's lower 24-bits should be notified at the devices identification label or in striking distance. It is recommended to include an area at the identification label for this purpose. Taking no use of this area implies that the lower 24-bits of the MAC are identical to the lower 24-bits of the SNR.

The devices MAC can be derived from the data visible at the identification label. Assuming the identification label carries no other notification, the above sample data implies that the devices default MAC is: 0x003C7E001B2D.

It is recommended to add a 2D-barcode to the identification label containing a string according to following format (*yxz* may denote the device-type):
“http://myurl.com/?TYP=SDMN-xyz&MID=003C7E&SNR=001B2D”

Label example:

Remark: scanning the label and documenting the device location during installation enables a convenient way to collect network configuration data and gain future access to devices manuals.

5.7 Network scan

Explicit identification and addressing of a SDMN device is enabled by device-type (MID + type-identifier), SNR and MAC. **Read-out of the identifiers via the network must be implemented.** Please also see: “14 SDMN - device identification summary“.

6 Line termination

A link network is based on the idea that a transmission line connects just two nodes. This is an ideal application for RS-485 based communication, cause the essential line termination can be included in the devices during manufacturing. So there is no need of any setup during installation. This results in ideal transmission line termination and an installation and operation without hassle.

7 Transmission processing

Receive and transmit of messages simultaneously at speeds of 115200 bit/s on multiple serial lines represents a high load for the processing unit if this has to be done by software. Modern processing architectures can handle these transfers by use of direct memory access (DMA). Using this technique the transmit of a message just represents a load of three, receive just a load of one interrupt routine to the processing unit - independent of the messages length. This way the transmission and reception of messages has only very limited influence on the processing load.

The processing unit can focus on message validation, the internal transport of messages to and from memory, re-transmit tasks and the application program.

8 Transmission control

Transmission line allocation by use of tokens or time slots is not implemented. Collisions happen and are tolerable.
Cause only two devices share a transmission line, collision detection and minimization is possible without producing to much overhead on a link.

Link activity is controlled by a state machine using 4 states:

- *IdleSetupState*
  - Link interface is initialized, the following state is *IdleState.*
- *IdleState*
  - Link interface is idle, receives data or is trying to enter *TransmitState*.
- *ReceivedState*
  - Is forced by interrupt (“idle line detection”). Happens when a message was received and the link interface is idle again. The following state is:
  1. *IdleSetupState*
     - when the received message contained a corrupted checksum (CRC).
     - when the message buffer is full, so the received message can not be stored.
     - when the received message was “sole acknowledge” (must not be acknowledged).
  2. *TransmitState*
     - when the receipt of the message must be acknowledged.
- TransmitState
  1. Entered instantly when a message with valid CRC was received to send an acknowledge. Transmit must be ongoing before *TxDelayACK* expires, see figure 1.
     - transmit of a message with the “ACK”-bit set when the message buffer is not empty.
     - transmit of a “sole acknowledge” message when the message buffer is empty.
  2. Entered when the message buffer contains a message to be transmitted.
     - AND the max. number of message re-transmits is not reached
     - AND no ongoing receive is detected
     - AND *TxDelay* is zero.
  - Supervised by a timer routine. If a timeout elapses, the link interface will be forced to *IdleState* via *IdleSetupState* by use of a watchdog handler. Exit of *TransmitState* is forced by interrupts (“transmit complete” or “idle line detection”), or transmit timeout.

three flags:

- *flagTxDelayACK*, indicating that an ongoing receive was detected during period *TxDelayACK.*
- *flagTxSoleAck*, indicating the type of message previously transmitted was “sole acknowledge”.
- *flagRxSetAck*, indicating that all messages received have to be stored in message buffer with the ack flag for this link set.

and one variable:

- *TxDelay*, the hold-off period in ms till the interface can enter transmit mode again. When it's value is different from zero, entering transmit mode is inhibited.

Some rules apply:

- The receive of all messages, except “sole acknowledge” must be acknowledged (see “15.1.1 ACK” for description of the acknowledge message format).
- After transmit of a message the link interface enters *IdleSetupState*. The value of *TxDelay* is set to infinity and an interrupt driven timer of 1 ms duration (*TxDelayACK*) is started.
  - in case the message previously transmitted was not "sole acknowledge" and an no ongoing receive is detected during this 1ms period, *flagTxDelayACK* is reset (indicating a previous collision).
  - in case the message previously transmitted was "sole acknowledge" *TxDelayACK* forces a break between successive transmits to enable the receiving link to detect discrete messages.

  After *TxDelayACK* expired *TxDelay* is set to zero.

- A link interface that transmitted a message and that could not detect idle link will instantly re-transmit the message, see figure 2.
- A link interface with *flagTxDelayACK* in reset state will not start a new transmission before an additional random delay called *TxDelayRND* expired. This is ensured by activating *TxDelay* for this period, see figure 3. This also applies for the first transmit after the nodes power up. The delay time *TxDelayRND* is calculated by use of a random number. The seed for the calculation of the random number is the unique MAC value. Delay time can be 0, 4, 8 or 12 ms. At a high probability both nodes connected by a link will start the next transmission at different times, so the conflict should be solved.

If the transmitting node receives no valid acknowledge message this may have several reasons:

- A collision on the transmission line occurred, cause the counterpart started a transmission at about the same time. A collision conflict will be dissolved by applying the above rules.
- The counterparts message buffer is already completely full.
- A counterpart does not exist. This topic will be described later, see "9.3 Open links".
- Damaged transmission line or link interface.

Legend	R=receive, T=transmit, acknowledge, collision, receive, transmit, *TxDelay*

Communication, no collision (figure 1):

Time	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
node A	R	R	T	T	T	T	T	T	T	T	R	R	R	R	R	R	R	R	R	R	R	R	R	T	T	T	R	R	R	R	R	R	R	R	R
node B	R	R	R	R	R	R	R	R	R	R	R	T	T	T	T	T	T	T	T	T	T	T	R	R	R	R	R	R	R	R	R	R	R	R	R
link	R	R	T	T	T	T	T	T	T	T	R	T	T	T	T	T	T	T	T	T	T	T	R	T	T	T	R	R	R	R	R	R	R	R	R

A transmits message (4-11), B transmits message with acknowledge (13-23), A transmits sole acknowledge (25-27).

Communication, collision, node can not detect idle link during *TxDelay* (figure 2):

Time	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
node A	R	R	T	T	T	T	T	T	T	T	R	R	R	R	T	T	T	T	T	T	T	T	R	R	R	R	R	R	R	R	R	R	R	R	T
node B	R	R	R	T	T	T	T	T	T	T	T	T	T	R	R	R	R	R	R	R	R	R	R	T	T	T	T	T	T	T	T	T	T	R	R
link	R	R	T	T	T	T	T	T	T	T	T	T	T	R	T	T	T	T	T	T	T	T	R	T	T	T	T	T	T	T	T	T	T	R	T

A transmits message (4-11), B transmits message (5-14), A can not detect idle link (12-14) and instantly re-transmits message (16-23), B re-transmits message with acknowledge (25-34), A transmits sole acknowledge (36-38), Hint: (37-38) not shown – see (26-27) of figure 1.

Communication, collision, no node detects ongoing receive during *TxDelayACK* (figure 3):

Time	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
node A	R	R	T	T	T	T	T	T	T	T	R	R	R	T	T	T	T	T	T	T	T	R	R	R	R	R	R	R	R	R	T	T	T	R	R
node B	R	R	R	T	T	T	T	T	T	T	R	R	R	R	R	R	R	R	R	R	R	R	T	T	T	T	T	T	T	R	R	R	R	R	R
link	R	R	T	T	T	T	T	T	T	T	R	R	R	T	T	T	T	T	T	T	T	R	T	T	T	T	T	T	T	R	T	T	T	R	R

A transmits message (4-11), B transmits message (5-11), both do not detect ongoing receive during *TxDelayACK*, A inserts extra random TxDelayRND (14), B inserts extra random TxDelayRND (14-16), A re-transmits message (15-22), B re-transmits message with acknowledge (24-30), A transmits sole acknowledge (32-34).

9 Message spreading and processing

This networking concept is based on the concept of forwarding and does not include routing.

Some methods to address nodes:

- by destination MAC
- by destination group
- by device-type

and two methods to suppress forwarding:

- HopLimit
- filter mechanism

are implemented into the protocol.

In contrast to message transmission that is mainly done by hardware, message spreading (the node internal transport of messages between links and memory) and message processing is done by software.

Re-transmit and transport of messages from the link input registers to memory and from memory to the link output registers, Message processing, -generation and -erase are memory based procedures under program control, depending on flags.

All messages include an attribute field called the HopLimit. Upon generation of the message the HopLimit is set to a programmable initial value. The size of the initial value depends on the network structure. If network topology is ring or meshed the initial value for HopLimit must be set during configuration by using the following formula:

- count of nodes that reside in the network minus 1.

Received and valid messages are stored in a ring buffer with space for 16 messages, called message buffer. Each entry contains following information:

- message (including payload, overhead and acknowledge)
- length of the message
- N ack flags ($A_1 \ldots A_N$), one flag for each link interface of the node
- processed flag (P)

The filter mechanism is a method to keep messages local (e.g. on a certain room, floor, building or machine). Each message includes 4 filter-flags named F0...F3. Programming of the nodes filter-mask register is optional during configuration of the network. By default the filter-mask register is set to all bits 0, so no filter is set.

For message spreading and processing some rules apply:

- the HopLimit inside a message is decremented after a link is passed and the message is processed (instantly after a message is received). Also see 15.3
- data being received from one link has to be transported to all other links:
  - except the messages destination address matches the nodes MAC
  - except the HopLimit is 0
  - except a filter-flag (F1...F4) set in the message matches a set bit (bitwise and) in the filter-mask register. Also see 15.2.1
- After successful transmit the associated ack flag (A_N) is set. Messages with all ack flags ($A_1 \ldots A_N$) and the processed flag (P) set are completely transmitted and processed and are deleted from the message buffer.

This implies that messages that are filtered (not transmitted) by a node are nevertheless processed by the node.

9.1 Meshed network anomaly

When the network has a meshed topology (or if two link interfaces of the same node are erroneously connected via a link) the above mentioned rules most times will not be sufficient. A message will be duplicated every time a branch (node with more than 2 link interfaces) is passed. At meshed networks this leads to multi-passing of identical messages through nodes coming from different directions, so called “routing loops”. This will result in an enormous rise of message traffic inside the network until the expired HopLimit stops the transmission of the message and it's copies. To avoid this situation messages have to have a unique message identifier (UID). When passing a node it is checked whether the UID is a member in a circular storage named message lock buffer (MLB). If the MLB contains the UID the message is erased from memory, otherwise the UID is stored in the MLB and the message is processed. This UID must be transmitted within the message or can be a hash value of the message itself with HopLimit and ACK bits masked out. The length of a suitable circular buffer is dependent from the arrangement of the meshes and the overall network topology and size. If the MAC in addition to a message counter is used to build the UID, additional 8 bytes (6+2) of overhead have to be transmitted per message. The use of the CRC with HopLimit and ACK bits masked out for UID will not work reliable, because of the high probability that two different messages have the same CRC. However, all of these approaches need a significant amount of memory for the MLB and also will take some computing resources. So restricting the use of the network protocol to none meshed or just clear meshed structures where the use of a HopLimit is obviously sufficient may be the better solution. Nodes including MLB and protocol including UID will not further be considered in the following paper.

9.2 Phantom messages

If the topology of the network is ring or meshed more than one message with the same content may be alive within one network. So the same message may reach a node more than once.
This will happen under following circumstances:

- topology is ring and the HopLimit is bigger than number of nodes minus one
- topology is meshed and simply realized with just HopLimit

If communication works in a satisfying way this should not be a big problem, except under this circumstances inexplicit commands like “toggle the output” or “goto next position” are used within messages. To avoid the problem the use of explicit messages like “output on”, “output off” or “goto position xyz” should be obligatory.

9.3 Open links

If more than 10 re-transmissions of the same message occurred, the associated ack flag (A_N) in the message buffer is set and the transmission link is set to inactive for a period of 15 seconds (*TxDelayBREAK*) by activating *TxDelay* and *flagRxSetAck* for this period.
To prevent message buffer overflow during this period:

- all messages received or generated by the node are stored in the message buffer with the ack flag (corresponding the open link) set
- the ack flag (corresponding the open link) is set at all messages already contained in the message buffer

This ensures that all messages contained in the message buffer (also these generated by the node itself) can be processed completely and afterwards can be erased from memory.
This behaviour may lead to a loss of messages for the subsidiary nodes connected via this link.

Receive of data will set the transmission link immediately to active mode again. The reason for not setting the inactive period to infinite is to enable nodes to start communication automatically after a re-connect of a physical broken link (e.g. during installation).

Hint: If lost of message is an issue, important messages have to be send with ReqResp flag enabled, so the sending node will know after time-out of the requested response, that the message did not arrive.

9.4 Message buffer overflow

Network overload may lead to message buffer overflow within a node. On buffer overflow a node clears it's message buffer to prevent network blockage and to ensure forwarding of the message buffer overflow message. All links are set to *IdleSetupState* and it's *TxDelay* to zero. The message received, leading to overflow, is not acknowledged.

This condition represents a loss of the messages for the node and the subsidiary nodes.

The event is notified by error counter, link status LED, transmit of the “Node_error_counters” message and terminal output error message. The occasion can be circumvented by keeping network traffic within compliant limits and by configuring devices (filter-flags, HopLimit, etc.) to appropriate values.

10 SDMN - group addressing

Adding SDMN devices to groups enables addressing groups instead of single devices (see “15.1.4 AdrType”). This leads to a drop in data traffic within the network and to an ease in operation of the system.

The addressing of groups is supported by providing a programmable group-mask (GM) within the none volatile memory of a node. The GM is 48-bits long, organized in 6 bytes, by default all bits of the GM are set to 0.

When destination group addressing is used the content of the GM inside the node and the content of the message address field are bitwise anded. If the result is equal to the address field the message addresses the node and has to be processed, otherwise the message just has to be spreaded.

The organization of the GM is not defined into main- or sub-groups and can vary between projects or applications.

In the following example msb is the leftmost bit.

Example: Imagine a 4 level building with a lamp in every room, 3 rooms on every level and an extra lamp on every corridor. You might want to switch all lamps simultaneously, all lamps on a level or all lamps in the corridors or just a single individual lamp.
In this case the GM could be organized like this:

- 4 bits (bits 7...4) for the building level 4...1
- one bit (bit 3) for the corridor
- 3 bits (bis 2...0) for the room-number 3...1

According to this scheme the lamp on the second level, room-number 3 has to be programmed to GM 00100100_b. The corridor lamp on the second level has to be programmed to 00101000_b

Group address 11111111b addresses all lamps, group address 00101111b addresses all lamps on level 2, address 11111000b addresses all lamps on the corridors, address 00100111b addresses all room lamps on level 2 and address 00100100b addresses the lamp at room-number 3 on the second level.

Group addressing has no influence on the performance or the load to be processed on a node. The challenge of group addressing is to have a well planed and organized GM structure and to implement GM handling into a intuitive configuration program.

11 Network traffic tracing

To indicate message flow two mechanisms are integrated.

11.1 Record counters

Device internal counters keep record of device status and data transmission to enable a rating of the device availability and transmission quality of the node and the overall network performance.

Record counters can be set to initial value by command. Counters, except buffered counters, lose content instantly on power-loss of the node. A value of zero of the “number of power on resets” indicates that buffered counters lost content since the last set command was received. Buffered counters keep content for a minimum period of 60 minutes when power supply fails.

When a record counter reaches it's maximum value further increment of the counter is stopped.

Following record counters have to be implemented:

- number of power on resets, cold reboot, (buffered)
- number of software resets, warm reboot (watchdog, RST-button, supply-low), (buffered)
- number of low voltage conditions, (buffered)
- number of CRC errors, implemented on every link interface
- number of lost of data due to open link, implemented on every link interface
- number of message buffer overflows
- number of messages received (without sole acknowledge messages and without messages containing CRC errors), implemented on every link interface
- number of messages transmitted (without sole acknowledge messages but including acknowledged and not acknowledged messages), implemented on every link interface

A read out of the record counters via the network must be implemented.

11.2 Link status LED's

The status of every link is visualized by a green LED. The LED, controlled by a state machine, is showing following events:

Event	Activity	Cycles	Period	On-Time	Pause - pre	Pause - post
Message received or transmitted	flashing	1	* 500 ms	10 ms	---	---
CRC error on receive	blinking	1	* 500 ms	200 ms	---	---
Transmit aborted due max. no. of transmits reached (open link)	blinking	2	500 ms	200 ms	1000 ms	1000 ms
Message buffer overflow	blinking	3	500 ms	200 ms	1000 ms	1000 ms

* minimum period

12 Device configuration

Configuration can be done in several ways:

- by sending messages via links containing configuration data
- by receiving input data via the USB based terminal interface of a node
- nodes may offer special hardware (like switches, jumpers or keys and display) to enable setup without using any external equipment at all

13 SDMN - device-property

SDMN devices have properties. A case sensitive character, optionally followed by a number, identifies a single property. A sequence of properties builds the device-property.

Character	Number	Property
R	n	RTC server (n: 0=crystal clock, 1=radio clock, 2=NTP)
T	n	Terminal interface (n: 0=via USB, 1=via RS-232)
V	-	Server
W	-	Web server functionality
L	n	“*Link network protocol via RS-485”* interfaces (n: 0, 2...4)
U	n	USB interface(s)
S	n	RS-232 interface(s)
E	n	wired Ethernet-Interface(s)
e	n	wireless Ethernet-Interface(s)

When number is omitted n is equal to 1.
Sample: the device-property for a device including 2 link interfaces and a USB based terminal interface is “L2UT0”.

14 SDMN - device identification summary

Following identifiers are mandatory and stored in a SDMN device:

Information	Example	Message-type	Storage location
Device-type	003C7E 0007CA	00001	Firmware
SNR	000000 001B2D	00001	Configuration memory
Device-property	“L2UT0“	00001	Firmware
Name of manufacturer	“MANUFACTURERname”	00004	Firmware
Name of product	“PRODUCTname”	00004	Firmware
Revision of hardware	1.0.3	00005	Configuration memory
Revision of firmware	2.53.7	00005	Firmware

For description of types and details see “5 SDMN - node identification“ and “17 SDMN - table of types“.

Access to device identification can be performed by message-types 00001, 00004 and 00005 or terminal interface. Lower 24 bits of MAC may be changed by use of message-type 00002 (MACsetup, see “16.5.2“) or terminal interface.

The 48-bit MAC used for addressing the device inside a network consists of:

- bits 24..47 are copied from device-type bits 24..47 (MID) and can not be altered
- bits 0..23 are copied by default from SNR bits 0..23 and may be altered, see “5.5“

Example: default MAC generated from above identifiers is 0x003C7E001B2D

For identification purposes each SDMN device must carry a label, see “5.6”.

15 SDMN - message frame

All messages except the *sole acknowledge* message consist of 16...64 bytes. They contain 0...48 bytes of payload, depending on the message-type.

A *sole acknowledge* message consists of just 3 bytes, the TAG0 and two CRC bytes.

Byte-Count								1								2	3	4	5	6	7	8	9	10	11	12	13	14	n = 0...48	15 + n	16 + n
Value	1	2	4	8	10	20	40	80	1	2	4	8	10	20	40	80	LSB	MSB	LSB					MSB	LSB	MSB				LSB	MSB
Bits	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Field	Tag0								Tag1								HopLimit		Address						Message-type		reserved		Data	CRC	
Name	ACK	ReqResp	Priority	AdrType0	AdrType1	reserved	reserved	reserved	F0	F1	F2	F3	reserved	reserved	reserved	reserved	HopLimitL	HopLimitH	Address0	Address1	Address2	Address3	Address4	Address5	MsgTypeL	MsgTypeH	reserved	reserved	Data	CrcL	CrcH

Bits and bytes marked “reserved” must not be used.

Remark: SDMN message frames transferred by protocols deviant from “*Link network protocol via RS-485*” may be enwrapped according to the particular requirements and may utilize included Tags and HopLimit differently.

15.1 Tag0

Tag0 contains 8 message flags.

15.1.1 ACK

Acknowledge has to be send after a message was received. Sending of ACK can be send in form of *sole acknowledge* message or may be included in a standard message frame.

Sole acknowledge message (3 bytes):

Description	Value
TAG0	0x01
CRC LSB	0x7E
CRC MSB	0x80

Standard 16...64 byte messages:

Description	Value
ACK	0x01
NAK	0x00

NAK is send when a message is transmitted and no outstanding acknowledge exists (this applies to all messages directly following the receive or transmit of a *sole acknowledge* message or to the first message transmitted).

15.1.2 ReqResp

Description	Value
Request a response message	0x02
No response expected	0x00

When the flag is set the source emitting the message directs the destination to send a response. The type of the response and the contained AdrType depend on the initiating message and may also depend on device setup.

Example1: device1 initiates device2 to do action1 (e.g. “go to position xyz”). After action1 finished, device1 will initiate action2 (e.g. “eject”). Action1 may last some time and it is mandatory to start action2 not before action1 finished. So device1 sends the message initiating action1 with ReqResp flag set. Upon receive of the message from device2 confirming that action1 finished, device1 initiates action2.

15.1.3 Priority

Description	Value
Message spreading with priority	0x04
Message spreading without priority	0x00

When the priority flag is set the message is prioritized when stored in the message buffer of a node. So it can pass messages that are already in the buffer.

15.1.4 AdrType

Defines the meaning of the 48-bits in the address field of the message.

Description	Value
Source MAC	0x00
Destination MAC	0x08
Destination group	0x10
Destination device-type	0x18

Messages including “Source MAC” are processed by all devices receiving the message. Messages containing other address types are processed just if addressing matches.
Further transport of message is not affected by address type, except address type is “Destination MAC” and address matches (in this case message has reached it's destination and must not be transferred any more).
For “Destination group” see “10 SDMN - group addressing“, for “Destination device-type” see “5.3 Device-type“.

15.2 Tag1

Tag1 contains 4 message flags.

15.2.1 Filter-flags

Description	Value
Message is global, no erase after entering node	0x00
Erase message after processing if flag F0 is set in node too.	0x01
Erase message after processing if flag F1 is set in node too.	0x02
Erase message after processing if flag F2 is set in node too.	0x04
Erase message after processing if flag F3 is set in node too.	0x08

These flags are for keeping messages local, see “9 Message spreading and processing”. Filter-flags do not change when passing a node. The setting of filter-flags in a new message generated by a node depends on the nodes “filter-flags” setting. Filter-flags included in a message are compared with the content of the local nodes filter-mask register (bitwise and), if result is unequal to 0 message transport is inhibited.
Example: By use of two flags, a message generated in a room can stay local in the room, and another message generated in the same room can stay local on the floor. The first message will be erased when trying to leave the room and the second when trying to leave the floor due to the settings in the filter-mask register of the nodes building the borders. In some ways this also can be done by setting an appropriate HopLimit, but when using the HopLimit this value has to be configured differently in every node.

15.3 HopLimit

Contains a 16-bit value. The HopLimit inside a message is decremented after a link is passed and the message is processed. If HopLimit is 0 the message will be erased from memory. Also see 9

15.4 Address

Contains 48-bit value, according to the definition by AdrType.

15.5 Message-type

Contains 16-bit value. The value defines the type of message, see “16 SDMN - messages”. (Reserved bytes following Message-type may be used in future specifications to expand the value to 24- or 32-bit.)

15.6 Data

Contains 0...48 bytes of data. Data contained depends on the type of message. See “16 SDMN - messages” and “17 SDMN - table of types”. If the message-type transmits data longer than 48 bytes, the first byte(s) of the data field are used for identification of the data fragment.

15.7 CRC

Contains 16-bit CRC checksum of the complete message excluding the last two bytes (CRC). Polynomial: x^16 + x^15 + x^2 + 1 (0xa001). Initial value: 0xFFFF. For sample C-code see “18.1 CRC calculation”.

16 SDMN - messages

Information inside an SDMN network is transported by messages. Three sorts of messages exist:

- request
- response
- transmit

Request messages are send to demand for information. *Response* messages contain information to meet demands. *Transmit* messages containing data or commands are send on event (e.g. input key pressed) or periodical (e.g. outdoors climate) to give information to the network.

16.1 Message-types and data-types

For interoperability of devices message-types are defined. Message-types are based on data-types. Both kind of types can be expanded on demand. To keep compatibility between devices expansion of types must be handled with care. Expansion of types must result in an upgrade of the table of types and it's associated version number, to keep interoperability of devices. See “17 SDMN - table of types“

16.2 Data encoding

The encoding of data inside a type follows the rule that the lsb is transmitted first. So at integer values the LSB is transmitted first and at floating point values the byte containing the sign is transmitted last.

16.3 Message class

Message-types classified “mandatory” must be supported by a device.

Message-types classified “recommended” should be supported by a device to enable comfortable handling.

Support of message-types classified “optional/specific” is device dependent. Every device may have special needs to support specific message-types to enable efficient communication.

See “17 SDMN - table of types” for assignment and classification.

16.4 Message sort

“Request”, “Response” and “Transmit” denote the sort of a message-type. Not all sorts of a message must be implemented on a device or message-type.

Messages of sort REQUEST apply for information transfer.

Messages of sort RESPONSE meet demand of information transfer.

Messages of sort TRANSMIT transfer information and commands.

16.5 Message-types

In the following find the usage and definition of existing message-types and sorts. For details see “17 SDMN - table of types“

16.5.1 DeviceIdentifier

00001	Request	Request	Response / Transmit
Purpose	Request device identifier	Request device identifier	Post device-type, SNR, MAC and device-property
Payload	none	none	39
ReqResp flag	set	set	reset
AdrType	Source MAC	Destination MAC or group	Source MAC

Identify devices attached to network. When message 00001 (Request including source MAC) is transferred all devices receiving the message will respond by sending message 00001 (Response). **Multiple responses containing the same source MAC imply address conflicts that must be solved by assigning unique MAC's** by use of message 00002 or terminal interface.

16.5.2 MACsetup

00002	Transmit
Purpose	Setup of MAC bits 0..23 via network
Payload	15
ReqResp flag	reset
AdrType	Destination MAC

Upon receive of message transmitted MAC bits 0..23 will be stored in configuration memory of device when following both conditions are true:

- device-type (see 5.3) transmitted and contained in device is identical
- SNR (see 5.4) transmitted and contained in device is identical
- device is in administration mode (see 16.5.23)

After MACsetup verifying the assignment of a unique MAC's by transmitting message 00001 (Request including source MAC) is recommended.

16.5.3 GMsetup

00003	Request	Response	Transmit
Purpose	Request group-mask	Post group-mask.	Setup group-mask
Payload	none	6	6
ReqResp flag	set	reset	reset
AdrType	Destination MAC	Source MAC	Destination MAC or group

Message of sort transmit is processed only when device is in administration mode (see 16.5.23).

16.5.4 DeviceLabels

00004	Request	Response / Transmit
Purpose	Request device labels	Post name of manufacturer (or it's URL) and name of product
Payload	none	48
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

16.5.5 DeviceRevisions

00005	Request	Response / Transmit
Purpose	Request device revisions	Post revision of hardware and firmware
Payload	none	6
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

16.5.6 NodeSetup

00006	Request	Response	Transmit
Purpose	Request node setup data.	Post node setup data.	Post node setup data.
Payload	none	4	4
ReqResp flag	set	reset	reset
AdrType	Destination MAC	Source MAC	Destination MAC

Setup of filter-flags, filter-mask register and HopLimit. Useful if node-setup must be archived or node must be cloned.
Message of sort transmit is processed only when device is in administration mode (see 16.5.23).

16.5.7 NodeErrors

00007	Request	Response / Transmit
Purpose	Request node errors	Post node errors.
Payload	none	42
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

16.5.8 NodeStatistics

00008	Request	Response / Transmit
Purpose	Request node statistics	Post node statistics.
Payload	none	32
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

16.5.9 DummyMessage

00009	Transmit
Purpose	Post dummy message, must not be processed by nodes, just transported. Produce network traffic for test purposes.
Payload	48
ReqResp flag	reset
AdrType	Source MAC

16.5.10 DeviceSetupBin

00010	Request	Response	Transmit
Purpose	Request device setup data frame	Post device setup data frame	Post device setup data frame
Payload	none	1...48	1...48
ReqResp flag	set	reset	reset
AdrType	Destination MAC	Source MAC	Destination MAC

Useful if device setup must be archived or device of same type must be cloned. Binary data, content see device/firmware documentation.
Message of sort transmit is processed only when device is in administration mode (see 16.5.23).

16.5.11 ClimateLocal

00011	Request	Response / Transmit
Purpose	Request climate local dataset.	Post climate local dataset.
Payload	none	18
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

16.5.12 DigitalInputEvent

00012	Transmit
Purpose	Post digital input event.
Payload	3
ReqResp flag	reset
AdrType	Source MAC

Post message upon actuation of digital input. Including input-number, click-type and event counter incremented by 1 on every event.

16.5.13 FileFirmware

00013	Transmit
Purpose	Transmit firmware-file to update devices/nodes.
Payload	48
ReqResp flag	reset
AdrType	Destination device-type or destination MAC

Sends the frame number followed by 44 data bytes. The first frame number to be transmitted is 0. The last frame (EOF) number to be transmitted is 0xFFFFFFFF, identifying the end of file.
Received message is only accepted when device is in administration mode (see 16.5.23).

16.5.14 SupplyVoltage

00014	Request	Response / Transmit
Purpose	Request supply voltage of device.	Post supply voltage of device.
Payload	none	4
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

Detection of supply voltage at point of load.

16.5.15 DOpwm

00015	Request	Response	Transmit
Purpose	Request DO status.	Post DO status.	Post DO status.
Payload	1	2	2
ReqResp flag	set	reset	reset
AdrType	Destination MAC	Source MAC	Destination MAC or group

Control digital PWM output.

16.5.16 ClimateOutdoors

00016	Transmit
Purpose	Post climate outdoors dataset.
Payload	18
ReqResp flag	reset
AdrType	Source MAC

When outdoors sensor is available and proper configured, this message is transmitted periodically, so request for this kind of data makes no sense.

16.5.17 AppData

00017	Request	Response/Transmit
Purpose	Request application data	Post application data
Payload	none	3...48
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

Post content (flag, text, data, unit, …) to external device. Content is application specific, so transmitter and receiver have to interpret payload the same way. Receiver must check if source MAC, application identifier and type inside payload match. To increase reliability receiving node may also check DeviceIdentifier of transmitter.

16.5.18 AnalogInputVoltage

00018	Request	Response / Transmit
Purpose	Request AI voltage.	Post AI voltage.
Payload	1	5
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

Read input voltage from analog input.

16.5.19 AnalogOutputVoltage

00019	Request	Response	Transmit
Purpose	Request AO voltage.	Post AO status.	Post DO status.
Payload	1	5	5
ReqResp flag	set	reset	reset
AdrType	Destination MAC	Source MAC	Destination MAC or group

Control analog output.

16.5.20 DigitalInputState1

00020	Request	Response / Transmit
Purpose	Request DI state.	Post DI state.
Payload	1	2
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

Read state of single digital input.

16.5.21 DigitalInputState8

00021	Request	Response / Transmit
Purpose	Request DI group state.	Post DI group state.
Payload	1	2
ReqResp flag	set	reset
AdrType	Destination MAC	Source MAC

Read state of digital input group (8 inputs, bit-coded).

16.5.22 DisplayContentDm2

00022	Transmit
Purpose	Post content to 2-digit dot matrix display.
Payload	7
ReqResp flag	reset
AdrType	Destination MAC or group

Control content of 2-digit dot matrix display on a remote resource (external device). Every content consists of a 2-digit header and 2-digit data followed by a break (no pixel set). Normally contents are shown in a continuous ascending order. Header and data are visible for a programmable period each, break period is programmable too. The number of the content succeeding (“next content index”) is programmable. Some content, depending on device and running application, may be fixed and can not be altered.
The number of contents that can be added depends on device and application. Contents added appear on display after the fixed contents. The number (1..255) of the content is always related to the added contents. Fixed contents, except the lowest order one (that can be forced or addressed by the “next content index” by using a value of 0), can not be addressed.
Adding a content copies the added contents index to the previous contents “next content index”.
Removing a content modifies the previous’s contents “next content index” to address the lowest fixed content.
Updating a content does not alter the previous contents “next content index”.
Forcing a content forces the “next content index” instantly to the value forced.
Devices showing received contents mark a content “out of date” or erase it from display if the message carrying the content is not received cyclical every 120 seconds, so a message carrying a content must be repeated within a period < 120 seconds.

16.5.23 AdminMode

00023	Transmit
Purpose	Enter or exit administration mode.
Payload	1..48
ReqResp flag	reset
AdrType	Destination device-type or destination MAC

To enter administrator mode transmitted password must match administrator password of receiving device. When transmitted password does not match exit of administrator mode is performed.

17 SDMN - table of types

The table shows all valid types and messages according to the specification. It carries a discrete version number to enable independent changes in specification and the table of types.

Link Network Protocol via RS-485 „Class“ definitions:

Class	Description
10	basic
20	unit
30	messages – mandatory
31	messages – recommended
32	messages – optional/specific

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
	10	0	8-bit signed integer, s8	00001	1			-128	127	-5
	10	0	8-bit unsigned integer, u8	00002	1			0	255	0x23
	10	0	8-bit character, ASCII coded, c8	00003	1			0x00	0x7F	7-bit ASCII code, 0x31==„1”, 0x41==„A”
	10	0	16-bit signed integer, s16	00004	2			-32768	32767	-3561
		1	Byte1, LSB	00002	1					
		2	Byte2, MSB	00002	1					
	10	0	16-bit unsigned integer, u16	00005	2			0	65535	0x1234
		1	Byte1, LSB	00002	1					
		2	Byte2, MSB	00002	1					
	10	0	32-bit signed integer, s32	00006	4			−2147483648	2147483647	-356178
		1	Byte1, LSB	00002	1					
		2	Byte2	00002	1					
		3	Byte3	00002	1					
		4	Byte4, MSB	00002	1					
	10	0	32-bit unsigned integer, u32	00007	4			0	4294967295	0x12345678
		1	Byte1, LSB	00002	1					
		2	Byte2	00002	1					
		3	Byte3	00002	1					
		4	Byte4, MSB	00002	1					
	10	0	64-bit unsigned integer, u64	00008	8			0	0xFFFFFFFF FFFFFFFF	0x1122334455667788
		1	Byte1, LSB	00002	1					
		2	Byte2	00002	1					

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
		3	Byte3	00002	1					
		4	Byte4	00002	1					
		5	Byte5	00002	1					
		6	Byte6	00002	1					
		7	Byte7	00002	1					
		8	Byte8, MSB	00002	1					
	10	0	32-bit floating point, f32	00009	4			$5{,}877 \cdot 10^{-39}$	$3{,}403 \cdot 10^{38}$	According IEEE 754, single
		1	Byte1, LSB	00002	1					
		2	Byte2	00002	1					
		3	Byte3	00002	1					
		4	Byte4, MSB, msb=sign	00002	1					
	10	0	64-bit floating point, f64	00010	8			$1{,}1125 \cdot 10^{-308}$	$1{,}798 \cdot 10^{308}$	According IEEE 754, double
		1	Byte1, LSB	00002	1					
		2	Byte2	00002	1					
		3	Byte3	00002	1					
		4	Byte4	00002	1					
		5	Byte5	00002	1					
		6	Byte6	00002	1					
		7	Byte7	00002	1					
		8	Byte8, MSB, msb=sign	00002	1					
	10	0	String	00011	1..48					„Hallo”, variable size
		1	Byte1, LSB	00003	1					H
		2	Byte2	00003	1					a
		..	Byte..	00003	1					
	20	0	Identifier6	00100	6	---				012345 6789AB
		1	Byte1, LSB	00002	1			0	255	0xAB

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
		2	Byte2	00002	1			0	255	0x89
		3	Byte3	00002	1			0	255	0x67
		4	Byte4	00002	1			0	255	0x45
		5	Byte5	00002	1			0	255	0x23
		6	Byte6, MSB	00002	1			0	255	0x01
	20	0	Revision of item	00101	3	---		0.0.0	255.255.255	2.53.7
		1	minor	00002	1			0	255	7
		2	mid	00002	1			0	255	53
		3	major	00002	1			0	255	2
	20	0	Temperature, °C	00102	4	°C	x 1000	- 273150	2147483647	83647 == 83,647°C
		1		00006	4					
	20	0	Humidity, relative, RH	00103	2	%	x 100	0	10000	7353 == 73,53 % RH
		1		00005	2					
	20	0	Humidity, absolute, AH	00104	4	g/m3	x 1000	0	588359	17285 == 17,285 g/m3
		1		00007	4					
	20	0	Pressure, Pa	00105	4	Pa	x 1	0	4294967295	101325 == 101325 Pa == 1013,25 hPa
		1		00007	4					
	20	0	Illuminance, lx	00106	4	lx	x 10000	0	4294967295	35000000 == 3500 lx
		1		00007	4					
	20	0	Digital out, PWM, 100 steps	00107	1	%	1	-1	100	-1 == overload, 0 == off, 100 == 100%
		1		00001	1					
	20	0	Digital in	00108	1	Byte	bit-coded	0	255	0x81 == bit7 and bit0 set

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
		1		00002	1					
	20	0	Voltage, mV, signed, 16-bit	00109	2	V	x 1000	-32768	32767	10000 == 10,000V
		1		00004	2					
	20	0	Voltage, mV, unsigned, 16-bit	00110	2	V	x 1000	0	65535	10000 == 10,000V
		1		00005	2					
	20	0	Voltage, mV, signed, 32-bit	00111	4	V	x 1000	-2147483648	2147483647	10000 == 10,000V
		1		00006	4					
	20	0	Voltage, mV, unsigned, 32-bit	00112	4	V	x 1000	0	4294967295	10000 == 10,000V
		1		00007	4					
	20	0	Seconds, s	00113	1	s	x 1	0	255	32 == 32 seconds
		1		00002	1					
	20	0	Minutes, min	00114	1	min	x 1	0	255	125 == 125 minutes
		1		00002	1					
	20	0	Hours, h	00115	1	h	x 1	0	255	5 == 5 hours
		1		00002	1					
	20	0	Identifier3	00116	3	---				6789AB
		1	Byte1, LSB	00002	1			0	255	0xAB
		2	Byte2	00002	1			0	255	0x89
		3	Byte3	00002	1			0	255	0x67
00001	30	0	DeviceIdentifier		0/39					Device identification
		1	Device-type	00100	6					003C7E 0007CA

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
		2	SNR	00100	6					000000 001B2D
		3	MAC, bits 0..23	00116	3			0	16777215	0x00001B2D
		4	Device-property	00011	24					„L2UT0 “
00002	30	0	MACsetup		15					Setup MAC bits 0..23 via network
		1	Device-type	00100	6					003C7E 0007CA
		2	SNR	00100	6					000000 001B2D
		3	MAC, bits 0..23	00116	3			0	16777215	MAC bits 0..23 to be stored
00003		0	GMsetup							Read /setup group-mask to enable group addressing
		1	Group-mask	00100	6					001D3F 00011A
00004	30	0	DeviceLabels		48					Device labels
		1	Name of manufacturer	00011	24					„MANUFACTURERname “, Text or URL
		2	Name of product	00011	24					„PRODUCTname “, Text
00005	30	0	DeviceRevisions		6					Device hard- firmware revisions
		1	Revision of hardware	00101	3					1.0.3
		2	Revision of firmware	00101	3					2.53.7
00006	30	0	NodeSetup		0/4					Node setup
		1	Filter-flags	00002	1			0	0x0F	Filter-flags
		2	Filter-mask register	00002	1			0	0x0F	Filter-mask register
		3	HopLimit	00005	2					HopLimit
00007	30	0	NodeErrors		0/42					Node errors
		1	Device, power on resets	00005	2					3, Number of power on resets
		2	Device, software resets	00005	2					2, Number of software resets
		3	Device, low voltage conditions	00005	2					5, Number of low voltage conditions
		4	Node CRC errors link1	00007	4					Number of errors

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
		5	Node CRC errors link2	00007	4					Number of errors
		6	Node CRC errors link3	00007	4					Number of errors
		7	Node CRC errors link4	00007	4					Number of errors
		8	Node open_link errors, link1	00007	4					Number of errors
		9	Node open_link errors, link2	00007	4					Number of errors
		10	Node open_link errors, link3	00007	4					Number of errors
		11	Node open_link errors, link4	00007	4					Number of errors
		12	Node message buffer overflows	00007	4					Number of errors
00008	31	0	NodeStatistics		0/32					Node traffic
		1	Node, messages received, link1	00007	4					Sum, excluding sole ACK messages
		2	Node, messages received, link2	00007	4					Sum, excluding sole ACK messages
		3	Node, messages received, link3	00007	4					Sum, excluding sole ACK messages
		4	Node, messages received, link4	00007	4					Sum, excluding sole ACK messages
		5	Node, messages transmitted, link1	00007	4					Sum, excluding sole ACK messages
		6	Node, messages transmitted, link2	00007	4					Sum, excluding sole ACK messages
		7	Node, messages transmitted, link3	00007	4					Sum, excluding sole ACK messages
		8	Node, messages transmitted, link4	00007	4					Sum, excluding sole ACK messages
00009	32	0	DummyMessage		48					Produce network traffic for test purposes
		1	Dummy string, 48-bytes	00011	48					„abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKL“
00010	30	0	DeviceSetupBin		0-48					Device setup data in binary format
		1	Data	00002	1-48					Setup data, binary, content see device/firmware documentation, max. 48 bytes (384 bit)
00011	32	0	ClimateLocal		0/18					Climate local
		1	Temperature, °C	00102	4					Sensor shielded from direct sunlight, corresponding to data 2
		2	Humidity, relative, RH	00103	2					Sensor shielded from direct sunlight, corresponding to data 1
		3	Pressure, Pa	00105	4					Optional data, maximal value if sensor not available

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
		4	Illuminance, lx	00106	4					Optional data, maximal value if sensor not available
		5	Temperature, °C	00102	4					Optional data, maximal value if sensor not available
00012	32	0	DigitalInputEvent		3					Digital input event
		1	Number	00002	1			1	255	Input number
		2	Type	00002	1			0	2	Type: 0x0 == click, 0x1==double click, 0x2 == long pressed
		3	Count	00002	1			0	255	Event counter, incremented by 1 on every event
00013	32	0	FileFirmware		48					Transmit firmware-file to update devices/nodes
		1	Frame number	00007	4					First frame ==0, last frame == 0xFFFFFFFF
		2	Data	00002	44					Firmware data
00014	30	0	SupplyVoltage		0/4					Supply voltage measured by local node
		1	Voltage, mV, signed, 32-bit	00111	4					
00015	32	0	DOpwm		1/2					Control digital PWM output
		1	Number	00002	1			1	255	Output number
		2	Digital out, PWM, 100 steps	00107	1					PWM data
00016	32	0	ClimateOutdoors		18					Climate outdoors, transmitted periodically
		1	Temperature, °C	00102	4					Sensor shielded from direct sunlight, corresponding to data 2
		2	Humidity, relative, RH	00103	2					Sensor shielded from direct sunlight, corresponding to data 1
		3	Pressure, Pa	00105	4					Optional data, maximal value if sensor not available
		4	Illuminance, lx	00106	4					Optional data, maximal value if sensor not available
		5	Temperature, °C	00102	4					Optional data, maximal value if sensor not available
00017	32	0	AppData		0/3-48					Application data provided for external device
		1	Application identifier	00002	1			0	63	Identifier of application currently running
		2	Type	00002	1			1	255	Data-set type-identifier
		3	Data	00002	1-46					Data-set (flag, text, data, unit, ...)

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
00018	32	0	AnalogInputVoltage		1/5					Analog input voltage
		1	Number	00002	1			1	255	Input number
		2	Voltage, mV, signed, 32-bit	00111	4					
00019	32	0	AnalogOutputVoltage		1/5					Analog output voltage
		1	Number	00002	1			1	255	Output number
		2	Voltage, mV, signed, 32-bit	00111	4					
00020	32	0	DigitalInputState1		1/2					State of single digital input
		1	Number	00002	1			1	255	Input number
		2	State	00002	1					Input state, 0x0 == reset, 0x1 == set
00021	32	0	DigitalInputState8		1/2					State of digital input group (group == 8 inputs)
		1	Number	00002	1			1	255	Input group number
		2	State	00108	1					Bit-coded, 0x41 == input 7 and input 1 are set
00022	32	0	DisplayContentDm2		8					Post content to 2-digit dot matrix display
		1	Remove / add / update / force	00002	1			0	1	Remove == 0, add == 1, update == 2, force == 3
		2	Content number	00002	1			0	255	Number of content, a value of 0 is valid for force only
		3	Header left digit	00002	1					Don’t care in case of remove or force
		4	Header right digit	00002	1					Don’t care in case of remove or force
		5	Data left digit	00002	1					Don’t care in case of remove or force
		6	Data right digit	00002	1					Don’t care in case of remove or force
		7	Display period	00002	1			0	255	100ms increments, don’t care in case of remove or force
		8	Break period	00002	1			0	255	100ms increments, don’t care in case of remove or force
		9	Next content index	00002	1			0	255	Number of next content to be shown, a value of 0 addresses the lowest fixed content, don’t care in case of remove or force
00023	30	0	AdminMode		1-48					Enter or exit administration mode

SDMN - table of types, Version 0.8.0

MsgType	Class	Order	Name / Description	DataType	Bytes	Unit	Factor	Min. value	Max. value	Example / Information
		1	Exit / enter	00002	1			0	1	Exit == 0, enter ==1
		2	Password	00011	0-47					„PASSWORD “, Text

18 Appendix

18.1 CRC calculation

The CRC used in the message frame is generated by use of the following C-code:

```
u16 CRCcalc(u8 *Buffer, u8 Bufferlength)
// calculate MODBUS compatible CRC-16 of Buffer with length Bufferlength
// Initial value: 0xFFFF
{
	u16 i, Val=0xFFFF;

	for (i = 0; i < Bufferlength; i++)
	{
		Val = crc16_update(Val, Buffer[i]);
	}
	return Val;
}

u16 crc16_update(u16 crc, u8 a)
// Optimized CRC-16 calculation.
// Polynomial: x^16 + x^15 + x^2 + 1 (0xa001)
// Initial value: 0xFFFF
// This CRC is used in disk-drive controllers and for MODBUS over serial line.
// see: http://www.lammertbies.nl/comm/info/crc-calculation.html
// see: http://www.nongnu.org/avr-libc/user-manual/group__util__crc.html,
util/crc16.h
{
	u8 i;

	crc ^= a;
	for (i = 0; i < 8; ++i)
	{
		if (crc & 1)
			crc = (crc >> 1) ^ 0xA001;
		else
			crc = (crc >> 1);
	}
	return crc;
}
```

18.2 SDMN - MID table

This list records all registered owners of a **M**anufacturer-**ID**entifier (MID) exceeding 0xFEFFFF.

See “ 5.1 Manufacturer-identifier“ for explanatory notes.

MID (Base 16)	Company name	Address
0xFF0000	SENG digitale Systeme	Alexanderstrasse 14, 73054 Eislingen, Germany

19 To Do's

This list is a collection of ideas, that may become requirements to be implemented in future versions of this specification or the table of types.

- add types and messages according to needs
- extend SDMN functionality (Web-Interface, SQL-database, device update procedure, …)
- store topics concerning SDMN in separate document
- implement some kind of user level mechanism within every node, so that processing of messages depends on authorization (e.g. message-types 00006 and 00010 should be performed in admin-mode only)
- implement optional encryption to link traffic (message frame) to prevent spy out and information manipulation. Infiltrating of message duplicates should be suppressed too.

20 Revision history

Document revision history:

Date	Revision	Changes
2012-08-13	0.9.0	Initial release
2013-03-04	0.9.1	48-bit CDN introduced, device classes revised
2013-03-26	0.9.2	Re-work of types and messages
2013-04-29	0.9.3	Changed coding of AdrTypes “Source MAC” and “Destination MAC”
2014-01-27	0.9.4	Changed open links (*TxDelayBREAK*) properties
2014-03-18	0.9.5	Changes concerning node identifiers and identification label content
2014-04-09	0.9.6	Bug fixes, copyright license fixed
2015-06-15	0.9.7	Added optional 2. temperature to message-type 00016
2016-10-31	0.9.8	Removed “data-point” message-types (00010...00012); added message-types 00010, 00011, 00012, 00017; revised message-types 00003, 00015, 00016
2017-02-03	0.9.9	Corrections
2017-07-10	0.10.0	Corrections and clarifications. Added message-types 00003, 00018, 00019, 00020, 00021, 00022, 00023, removed message 00002, re-ordered messages 00001...00003, revised node identification, revised message-types 00006 and 00010, added SDMN concept description, changed open link (9.3) behaviour, changed message buffer size to 16.
2017-11-21	0.10.1	Added insertion of *TxDelayACK* after transmit of "sole acknowledge" message to specification. Clarified chapter 8 “Transmission control“.