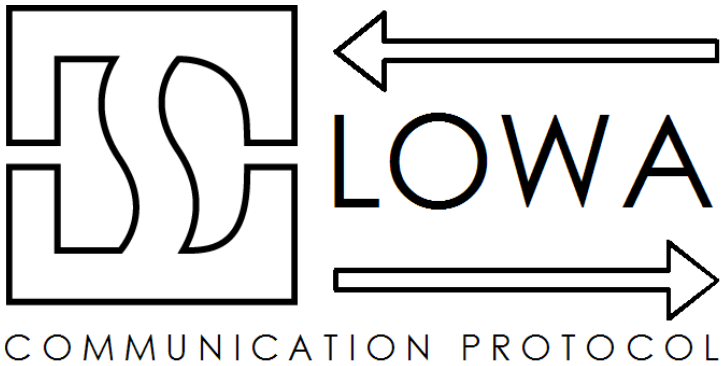
"Manual"

LOWA DIGI SENS protocol

An implementation guide for programmers



Overview

This document is intended for the people that are in charge of the implementation of the interface between a computer and a DIGI SENS LOWA system

Document Number

K321E-06

Referenced Documents

Revision history

19.05.2008	Emanuel Masel	Created
08.10.2008	Philipp Schultz	Published Version 02
15.10.2008	Emanuel Masel	Released Version 02
28.05.2013	Philipp Zaugg	Released Version 03
19.11.2018	Dominique Spack	Version 04
03.08.2020	Emanuel Masel	Released version 05, add the 'E' error.
21.04.21	Emanuel Masel	Add the functions "get all weight" and "set baudrate"

Filename

K321e-06_lowa_protocol.odt

Copyright

Table of Contents

1 Introduction

This document delivers all information needed to implement the communication between a computer and the DIGI SENS MUX.

The intended audience consist of programmers and project engineers that are in charge of realising the interface. Except for experience in interface protocols and common technical skills, no special previous knowledge is required.

The document starts with a chapter collecting most general information you need for a good start.

Chapter 3 "Hardware connection" shows you how to mount your system. It does also explain all details about powering and communication. This is the first step to have a running system.

You may start to implement the protocol (see "The LOWA DIGI SENS Protocol" in chapter 4) with the commands "gw" or "ag".

This document does not replace a workshop and a contact with DIGI SENS. Please feel free to contact us for any question.

2 General information

2.1 Configuration management

By implementing the LOWA DIGI SENS protocol, you are bound to implement future changes.

As in all protocols there is a trade off between keeping it compatible and extend the functionality. Since The MUX hardware has a limited capacity, we are not able to implement new commands and keep the old versions of the same functionality running.
On the set of commands described in this document we are very careful with such changes.

In the eventuality we have to change the behaviour of a command, we will send you an information saying "from date XYZ, the command to set the calibration factor behave like this...". If you wish so, you may have an early prototype of the new MUX to test your implementation.

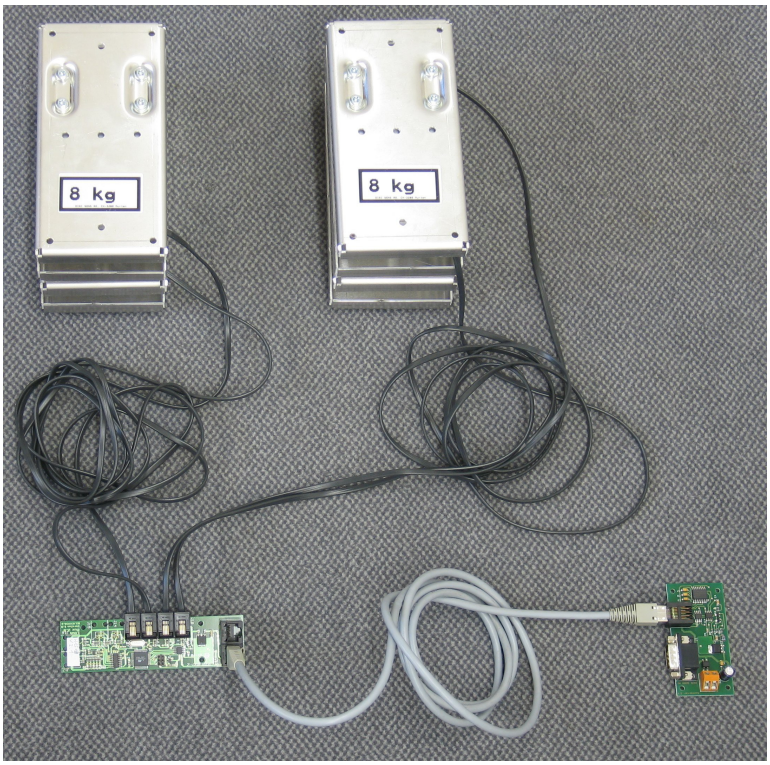
> ✎ The indications comprised in this manual are applying to all MUX software that where delivered since the end of 2007.
> Earlier versions may slightly differ.

2.2 Person that should be informed of changes

Configuration management is not only a question of your software having the capability to handle new commands, but also a question of knowing that there are new commands.
Please do therefore send an email to docs@digisens.ch containing the full contact information of the person that should be informed. Please do also make sure to send another email to the same address if this person changes.

2.3 Test-kit

Depending on the project set-up, you will probably get this manual during an implementation project.



An example of a test-kit

This manual gets along with a test-kit that is composed of

- two or more scales[1]
- one MUX with the software used in your project
- an e-nventory CONVERTER
- a cable to connect the MUX and CONVERTER

This test-kit gives you all you need to implement the interface.

1 If your test-set is composed of a MUX and separated scales, you should connect the scales to the MUX according to the stickers on the cables because the MUX was programmed with the calibration parameters of the scale.

2.4 Workshop and Assistance

This manual does also come along with a workshop to start the project. This workshop covers following topics:

- general architecture of e-nventory
- the LOWA interface
- do's and don'ts on interface level
- project steps

We do also use this workshop to talk about the details around like:

- Planning a site
- Mounting the components
- Measurement tolerances
- Is there any maintenance needed?
- What to monitor
- How to react on errors

You will also get the contact details of people that will assist you during that workshop.

2.5 e-nventory BOEXLI

DIGI SENS provides a tool that can issue most maintenance commands to the scales. Furthermore it is a good tool to test scales.

The "e-nventory BOEXLI" is equipped with an RJ-45 connector to connect the LOWA bus on one side. On the other side there is a DB9 connector that allows to connect to the serial port of a PC. The BOEXLI is also equipped with a power supply that feeds the bus with power.

The e-nventory BOEXLI

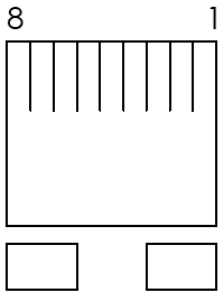
Although it is not mandatory, we do recommend you to have one e-nventory BOEXLI as a backup or if you fail to find out the reason of a malfunction.

3 Hardware connection

3.1 Connecting directly to MUX

If you would like to connect directly to the MUX, you need an RS485 capable device.

Furthermore you need a 12 V DC power source (minimum 7 V DC at the MUX entry).



1. RS485 (A)
2. RS485 (B)
3. not used
4. not used
5. GND
6. GND
7. +12 V
8. +12 V

The RS485 RJ45 pins

3.2 Connecting with an e-nventory CONVERTER

The e-nventory CONVERTER is an adapter that was designed to connect MUXes to an RS232 interface (typically a PC during development).

On the PC side you may connect the converter using a 1:1 (strait) serial cable with DB9 plugs.

You may connect the converter to any AC or DC power source between 9 V and 16 V. The polarity is not important.

3.3 Communication parameters

The MUXes are communicating with the following parameters :

- 9600 bauds
- 8 bit characters
- no parity
- one stopbit
- no flow control
- LSB first

4 The LOWA DIGI SENS Protocol

The "LOWA DIGI SENS Protocol" is a serial half-duplex ASCII protocol. There is no collision control.

> Your device will therefore act as a BUS-Master, meaning you are responsible to avoid collisions by sending one message and waiting for the answer before sending the next message.

We have defined this protocol and are using it for various applications including:

- WI-Systems on trucks
- Other WI-Systems
- DIGI SENS logistic systems
- DIGI SENS counting systems

You may use the commands published in this document as long as you

- use it with DIGI SENS components
- implement modifications we could make in the future
- give us feedback about problems or improvements

> The protocol published here under is a subset of the commands that are used in all applications and in the production process. Since some commands are amended or created weekly we leave the commands we feel you do not need unpublished.
>
> This will save you a lot of code maintenance and deployment time.

4.1 Device ID

Since there can be multiple devices on the bus, most of the commands use an address to assign the command to one of them. There are two different ways to address a device. Both addressing modes got the same protocol basis.

4.1.1 Standard address mode

In this mode every message starts with the character "@". In the standard mode the ID of every device needs to be set by the user. The ID (address) is a value from 000 to 999.

4.1.2 Extended address mode

If the message starts with the character "#" the device uses the manufacturer set unique ID. This ID can't be changed by the user. The ID got a length of 16 characters.

✎ Please consider that the extended addressing mode is not implemented in every device. Please contact DIGI SENS for further information.

4.2 Message syntax

Messages are case sensitive printable ASCII codes. The messages are beginning with the character "@" or "#". Every message ends with the character "CR" (carriage return 0x0D).

✎ Any following "LF" (line feed 0x0A) has to be ignored.

4.2.1 Question

LL = length

CM = command (usually lowercase)

CC = checksum (see below for formula)

Byte #0	Byte #1-2	Byte #3-4	X	N-2 – N-1	N
@	LL	CM	data	CC	"CR"

4.2.2 Answer

LL = length

CC = checksum (see below for formula)

Byte #0	Byte #1-2	X	N-2 – N-1	N
@	LL	data	CC	"CR"

4.3 Calculating the checksum

The checksum is a XOR on all previous bytes (including "@") modulo 0xFF, in Hexadecimal ASCII.

Examples:

"@09sz1230"

```
0x40 XOR 0x30 XOR 0x39 XOR 0x73 XOR 0x7A XOR 0x31 XOR 0x32 XOR 0x33 XOR
0x30 = 0x40
```

=> Complete command = "@09sz123040"

4.4 Ordinary commands

4.4.1 Request a single weight

This command is used to get a weight value.

Depending on the application this value is rounded to a reasonable resolution value that is programmed in the MUX.

> ✎ This first rounding does not prevent you from applying a second rounding that is reasonable for your implementation.

FUNCTION	COMMAND	Response from Scale
Get Weight Result is rounded to the resolution	@|LL|"gw"|nnn|h|CC|CR LL = "09" nnn = MUX-ID h = channel [one ASCII char]	@|LL|swwwwwwwwx|CC|CR LL = "13" s = sign (space or -) wwwwwwww = weight in kg.g (pad with leading 0's includes decimal point) x= Status: ● "M" = in Motion ● "C" = not connected ● "E" = eeprom error (4W-MUX only) ● Space = OK
Example:	@09gw123059 get the weight of MUX with the ID 123 channel 0	@13 0002.130 5C 2.130 kg

FUNCTION	COMMAND	Response from Scale
Get Weight Result is rounded to the resolution	#|LL|"gw"| nnnnnnnnnnnnnnnn|h|CC| CR LL = "22" nnnnnnnnnnnnnnnn = MUX-ID h = channel [one ASCII char]	#|LL|swwwwwwwwx|CC|CR LL = "13" s = sign (space or -) wwwwwwww = weight in kg.g (pad with leading 0's includes decimal point) x= Status: ● "M" = in Motion ● "C" = not connected ● "E" = eeprom error (4W-MUX only) ● Space = OK
Example:	#22gw12345678901234560 05 get the weight of MUX with the ID 1234567890123456 channel 0	#13 0002.130 3F 2.130 kg

✎ On palettes or differential-systems, this command, sent to any of the channel of a subsystem, will always give the same response:

The load on the whole subsystems

4.4.2 Request all the weight of the shelf

FUNCTION	COMMAND	Response from Scale
Get All Weight Result is rounded to the resolution	@|LL|"gl"|nnn|CC|CR LL = "08" nnn = MUX-ID	@|LL|swwwwwwwwx|CC|CR LL = "3+11*scale number" s = sign (space or -) wwwwwwww = weight in kg.g (pad with leading 0's includes decimal point) x= Status: ● "M" = in Motion ● "C" = not connected ● "E" = eeprom error (4W-MUX only) ● Space = OK
Example:	@08gl00172 get the weight of MUX with the ID 001	@91-00005.507E 00000.000C 00000.000C 00000.000C 00027.738 -00273.150C-00273.150C-00273.150C21 -5.507 kg with error E

		0.0 with error C 0.0 with error C 27.738 without error -273.15 with error C -273.15 with error C -273.15 with error C

FUNCTION	COMMAND	Response from Scale
Get All Weight Result is rounded to the resolution	#|LL|"gl"| nnnnnnnnnnnnnnnn|CC|CR LL = "21" nnn = MUX-ID	#|LL|swwwwwwwwx|CC|CR LL = "3+11*scale number" s = sign (space or -) wwwwwwww = weight in kg.g (pad with leading 0's includes decimal point) x= Status: ● "M" = in Motion ● "C" = not connected ● "E" = eeprom error (4W-MUX only) ● Space = OK
Example:	#08gl123456789012345626 get the weight of MUX with the ID 001	#91-00005.507E 00000.000C 00000.000C 00000.000C 00027.738 -00273.150C-00273.150C-00273.150C42 -5.507 kg with error E 0.0 with error C 0.0 with error C 27.738 without error -273.15 with error C -273.15 with error C -273.15 with error C

4.5 Zeroing

4.5.1 Zero scale

This command zeroes a scale. It should be used when the scale is empty.

FUNCTION	COMMAND	Response from Scale
Scale Zero	@|LL|"sz"|nnn|h|CC|CR LL = "09" nnn = MUX-ID h = channel [one ASCII char]	@|LL|"OK"|CC|CR LL = "05" CC = "01" "OK" = success, no answer if not OK
Example:	@09sz123040 Zero the scale channel 0 of the MUX with the ID 123	@05OK41 OK

FUNCTION	COMMAND	Response from Scale
Scale Zero	#|LL|"sz"| nnnnnnnnnnnnnnnn|h|CC| CR LL = "22" nnnnnnnnnnnnnnnn = MUX-ID h = channel [one ASCII char]	#|LL|"OK"|CC|CR LL = "05" CC = "22" "OK" = success, no answer if not OK
Example:	#22sz12345678901234560 1C Zero the scale channel 0 of the MUX with the ID 1234567890123456	#05OK22 OK

On differential-systems, this command should be issued to one of the channels of a subsystem. It will then zero the whole subsystem. The zero command should be issued when the scale is mounted and empty. It should be repeated when modifications get made to the parts under or over the scale (for example welding new parts, moving the scale).

This command is writing values on the MUX.

Since written values are stored in permanent memory chips that have a limited number of write cycles (ours are specified to have an endurance of 100'000 write cycles in average).
Once this limit is reached, the MUX will behave in a strange manner and must be replaced.

As a programmer you are responsible to create a software that will in no circumstances reach that limit during the whole lifetime of the MUX (some DIGI SENS systems are running since 1994).

As a consequence of the remark above, this command should not be used as a "tare weight" during normal operations. If a "tare weight" has to be implemented it should be done in the computer.

4.6 Addressing commands

If you are not using an "e-nventory BOEXLI" (the service tool provided by DIGI SENS) you will probably want to look up or set addresses from your software.

4.6.1 Get the address of the MUX

This command is getting the address of a MUX.

FUNCTION	COMMAND	Response from MUX
Address get	@|LL|"ag"|CC|CR LL = "05"	@|LL|nnn|CC|CR LL = "06" nnn = MUX-ID
Example:	@05ag43 Get the ID of the connected MUX (broadcast)	@060087E The ID is "008"

FUNCTION	COMMAND	Response from MUX
Address get	#|LL|"ag"|CC|CR LL = "05"	#|LL|nnnnnnnnnnnnnnnn|CC|CR LL = "19" nnnnnnnnnnnnnnn = MUX-ID
Example:	#05ag20 Get the ID of the connected MUX (broadcast)	#1912345678901234562D The ID is "1234567890123456"

Since this command is a sort of broadcast, it makes no sense – **and is therefore forbidden** – to issue it when more than one MUX is connected.

If you do so, you will probably

- get no answer (good)
- get an answer from the fastest MUX (bad, because it could be any of them and it might not always be the same one answering)
- get a mix of answers that gives a command that will execute something on the network (very bad, but also very unlikely)

4.6.2 Set the address of the MUX

This command is setting the address of a MUX.

FUNCTION	COMMAND	Response from MUX
Address set	@|LL|"as"|nnn|CC|CR LL = "08"	@|LL|nnn|CC|CR LL = "06" nnn = MUX-ID
Example:	@08as00862 Set the ID of the connected MUX (broadcast) to "008"	@060087E The ID is now "008"

The remark about "broadcasting" also applies here (see 4.6.1 "Get the address of the MUX").

> ✎ This command is writing values on the MUX the remark on page 17 about "limited number of write cycles" applies.

4.7 Software and revision commands

Every single MUX is programmed with a software that is related to the hardware (number of ports, release) and to the application (truck, silo, ...). This is what we call the model number.

Furthermore each software has different releases that are related to evolutions in the software.

The combination of software version and release is giving the command support, therefore the two following commands are essential to implement configuration control on the software side (see 2.1 "Configuration management").

> ✎ The result of these commands is probably a good information to be added to "installation report", "calibration report" and other documents related to installation and maintenance.

4.7.1 Get model number

Gets the model number.

FUNCTION	COMMAND	Response from MUX
Get model	@|LL|"gm"|nnn|CC|CR LL = "08" nnn = MUX-ID	@|LL|mmmmm|CC|CR LL = "08" mmmmm = model number
Example:	@08gm00775 Get the model number of MUX with the ID "007"	@08H110303 It is a "H1103"

FUNCTION	COMMAND	Response from MUX
Get model	#|LL|"gm"| nnnnnnnnnnnnnnnn|CC|CR LL = "21" nnnnnnnnnnnnnnnn = MUX-ID	#|LL|mmmmm|CC|CR LL = "08" mmmmm = model number
Example:	#21gm12345678901234562C Get the model number of MUX with the ID "1234567890123456"	#08H110360 It is a "H1103"

4.7.2 Get revision

Gets the revision.

FUNCTION	COMMAND	Response from MUX
Get revision	@|LL|"gr"|nnn|CC|CR LL = "08" nnn = MUX-ID	@|LL|mmm|CC|CR LL = "08" mmm = revision number
Example:	@08gr1016D Get the revision of the MUX with the ID "101"	@062.16B It is revision "2.1"

FUNCTION	COMMAND	Response from MUX
Get revision	#|LL|"gr"| nnnnnnnnnnnnnnn|CC|CR LL = "21" nnnnnnnnnnnnnnn = MUX-ID	#|LL|mmm|CC|CR LL = "08" mmm = revision number
Example:	#21gr123456789012345633 Get the revision of the MUX with the ID "1234567890123456"	#062.108 It is revision "2.1"

4.8 Analysis commands

4.8.1 Get data

This command is used to get the raw data from the MUX. It is mainly used to analyse frequencies for maintenance purposes.

This command can retrieve either the raw frequency of each sensor or the calibrated weight[2] of a scale.

> ✎ In palettes or differential-systems "get data weight" on a channel of a subsystem will always be the weight for the whole subsystem.

FUNCTION	COMMAND	Response from Scale
Get data Result is rounded to the resolution	@|LL|"gd"|nnn|h|k|CC|CR LL = "10" nnn = MUX-ID h = channel [one ASCII char] k = 0 for weight 1 for frequency	@|LL|swwwwwwwwwx|CC|CR LL = "14" s = sign (space or -) wwwwwwwww = weight in kg.g or frequency in Hz.mHz (pad with leading 0's includes decimal point) x= Status: ● "M" = in Motion ● "C" = not connected ● "E" = eeprom error (4W-MUX only) ● Space = OK
Example:	@10gd1230173 get the frequency of MUX with the ID 123 channel 0	@14 14000.000 6E 14000 Hz

2 The weight based on the calibration parameter that are stored in the MUX

FUNCTION	COMMAND	Response from Scale
Get data Result is rounded to the resolution	#|LL|"gd"| nnnnnnnnnnnnnnnn|h|k| CC|CR LL = "23" nnnnnnnnnnnnnnnn = MUX-ID h = channel [one ASCII char] k = 0 for weight 1 for frequency	#|LL|swwwwwwwwwx|CC|CR LL = "14" s = sign (space or -) wwwwwwwww = weight in kg.g or frequency in Hz.mHz (pad with leading 0's includes decimal point) x= Status: ● "M" = in Motion ● "C" = not connected ● "E" = eeprom error (4W-MUX only) ● Space = OK
Example:	#23gd12345678901234560126 get the frequency of MUX with the ID 1234567890123456 channel 0	#14 14000.000 0D 14000 Hz

4.9 Configuration command

4.9.1 Set baudrate

This command allow you to change the communication speed with the mux.
The response is given with the new speed.
The speed range are 9600 to 115200 with 9600 step.

FUNCTION	COMMAND	Response from Scale
Set baudrate	@|LL|"br"|nnn|aaaaaa|CC|CR LL = "14" nnn = MUX-ID a = bus speed with leading zero [one ASCII char]	@|LL|"OK"|CC|CR LL = "05" CC = "01" "OK" = success, no answer if not OK
Example:	@14br0010384006B Set baudrate to 38400 of the mux with the ID 01	@05OK41 OK

FUNCTION	COMMAND	Response from Scale
Set baudrate	#|LL|"br"|nnnnnnnnnnnnnnn|aaaaaa|CC|CR LL = "27" nnnnnnnnnnnnnnn = MUX-ID aaaaaa = bus speed with leading zero [one ASCII char]	#|LL|"OK"|CC|CR LL = "05" CC = "01" "OK" = success, no answer if not OK
Example:	#27br12345678901234560384003F Set baudrate to 38400 of the mux with the ID 1234567890123456	#05OK22 OK

4.10 Errors

4.10.1 OK

The MUX is programmed to be as failure safe as possible, meaning it will try to detect errors and report them using the error flag.
Doing so it will mostly indicate a value although there is an error. This is implemented like this because we feel there may be situations in your

application where we feel it is an error-state but you know that this situation is not relevant to you.

> ✎ In other words your application is responsible to monitor the error-states and to decide to or not to display a weight when an error-state (for example "motion") is on.

4.10.2 Motion

A scale is considered to be in motion when the variation of the measured values to the previous one exceeds a certain motion threshold.

In that case the MUX will send its current measured weight (that will not be within measurement tolerances) and the "M" flag.

This is often the sign, that somebody is taking part off the scale at this moment. If the error persists there might be something causing vibrations or other disturbances next to the scale.

> ✎ We imagine your software could repeat the request a bit later (a second or so). As long the error persists, no value should be displayed.

4.10.3 Not connected

If the MUX is not able to communicate with a single sensor, the response will have the "C" flag.

This is the sign, that something is wrong between the MUX and the sensor. Such errors may occur in some random patterns. If the error occurs often or permanently this is a sign that the component is broken and should be fixed by a service engineer.

> ✎ We imagine your software could repeat the request a bit later. As long the error persists, no value should be displayed.

4.10.4 EEPROM error

The 4W-MUX read the EEPROM of a 4W cell. If the reading of this EEPROM failed. The ‘E’ Flag is set. This means that MUX use default calibration values instead of the one in the cells, so the weight are not correct.

This sign can mean that the cell used are not a 4W cells or the are a connection problem.

> ✎ The EEPROM are read on the 4W-MUX startup, if you plug a cell after the startup, the EEPROM are not read. You just need to restart the 4W-MUX by unplugging and plug again the power supply of the MUX.

4.10.5 Other errors

There are no other error states by now. We are working on some new error states and will publish them later.

> ✎ We imagine your software could handle other errors and may pipe the unknown error code in a message like "unknown error 'X' sent by MUX 'XXX' for channel 'Y' "

4.11 Timing considerations

Following our experience reasonable polling frequency of one single scale in a LOWA system is above 1 second.

The maximum speed on multiple scales depends on the parameters explained here under.

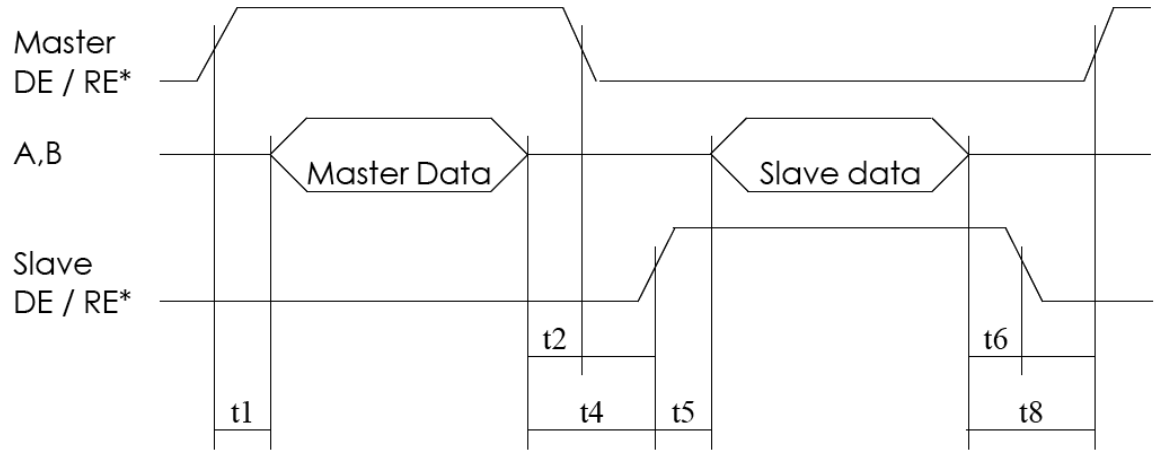
4.11.1 Measurement interval

On the sensor side, the MUX measures new values about every 200 ms (milliseconds) for each channel. Polling the values at a higher frequency make little sense but gives out the last measured value.

> ✎ Remember that if you use multiple scales to measure one item (for example one bin on two scales or the legs of the same silo) you should poll the scales that are measuring one item "at the same moment". Doing this you avoid summing weight that are representing a different situation.
> In extend you may also apply an error that is happening to one scale of the group to all of them. In other words if one scale says "motion" you should re-ask the other scales for their new weight and make the sum once none of the scales has a motion.

4.11.2 Reaction time

The usual reaction time of a MUX are comprised between 5 and 50 ms. If there is no answer you may repeat the request after a certain time out.



DE : Data enable (Sender Enable)

RE : Read enable (Receiver Enable)

Timing scheme

	min [ms]	max [ms]
t1	5	-
t2	2	-
t4	2.5	50
t5	5	-
t6	2	-
t8	2.5	-

4.11.3 Message transmission time

Since the messages are exchanged at a rather low speed, the transmission time of the message is relevant.

The formula to calculate it is :

```
duration = transmitted bytes x bits per byte / baudrate
```

Example: longest possible message = 105 chars

```
105 x 10 / 9600 = 0.109 [s]
```

Why 10 bits ?

1 start bit + 8 data bit + 1 stop bit

5 Considerations about the implementation

5.1 Power supply

Like every electronic component the MUX and the sensors is heating up its environment. This characteristic is rarely a problem for overheating.

Due to the extreme sensitivity of the sensors it could influence the measurements in the seconds following start-up.

Please discuss this issue with DIGI SENS if you plan not to power the sensors all the time.

5.2 Logging

Assuming the risk to be banal we feel, it could be useful to log events like

- start-up
- shutdown or power loss
- normal weighing processes (if they can be recognised)
- strange weight value (for example negative ones or overloads)
- error states
- debug value of frequencies and weights with a delay that can be set up
- ...

This greatly improves the ability to maintain such systems

5.3 Offset and span correction

Depending on the type of application you are implementing, you may either use factory calibrated components (like SHELVES, PALLETS) where you only need to zero or on-site calibrated devices.

In both cases it might be of interest to implement an additional offset a span correction in your system.

This gives you an easy way to influence the values without any change in the MUX.

Offset and span correction

In this example with exaggerated values, the offset is -1000, the span factor 20%.

6 Conclusion

Using this document you should be able to implement a system that is able to communicate with a DIGI SENS LOWA-system to get the weights and do all ordinary maintenance tasks (that does not require special tools/knowledge).

We hope that this document helped you. We would also appreciate to hear your comments applause or hoots at docs@digisens.ch. To help us improve the incriminated document please tell us also the document number and revision date.

